Titre: Title:	Algorithme de recherche tabou pour la planification optimale d'une campagne marketing sur les moteurs de recherche
Auteur: Author:	Mehdi Jaoua
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Jaoua, M. (2014). Algorithme de recherche tabou pour la planification optimale d'une campagne marketing sur les moteurs de recherche [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1503/

UNIVERSITÉ DE MONTRÉAL

ALGORITHME DE RECHERCHE TABOU POUR LA PLANIFICATION OPTIMALE D'UNE CAMPAGNE MARKETING SUR LES MOTEURS DE RECHERCHE

MEHDI JAOUA

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INDUSTRIEL)

AOÛT 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ALGORITHME DE RECHERCHE TABOU POUR LA PLANIFICATION OPTIMALE D'UNE CAMPAGNE MARKETING SUR LES MOTEURS DE RECHERCHE

présenté par : JAOUA Mehdi

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. LE DIGABEL Sébastien, Ph.D., président

M. GAMACHE Michel, Ph.D., membre et directeur de recherche

M. HERTZ Alain, Doct. ès Sc., membre et codirecteur de recherche

M. ADJENGUE Luc, Ph.D., membre

DÉDICACE

À ma famille et tous mes amis.

REMERCIEMENTS

D’abord, je tiens à remercier mon directeur de recherche Michel Gamache et mon codirecteur de recherche Alain Hertz qui m’ont permis, grâce à leur grande expertise, de mener à bien tout ce projet de maîtrise. C’est aussi grâce à mes professeurs que l’excellente relation de collaboration entre l’École Polytechnique de Montréal et Acquisio, m’a permis d’appliquer et d’approfondir mes connaissances dans le domaine de l’optimisation.

Je remercie également toute l’agréable équipe d’Acquisio en commençant par Richard Couture cofondateur et directeur principal des produits, pour m’avoir accueilli au sein de son équipe et pour tous les moyens nécessaires qu’il a mis à ma disposition pour le bon déroulement de mon stage. Ensuite, une reconnaissance très particulière envers Sandrine Paroz pour sa disponibilité et pour tout l’encadrement qu’elle m’a fourni grâce à sa grande expertise dans l’optimisation des campagnes marketing sur Internet. Je tiens aussi à remercier Mohamed Oussara qui n’a pas hésité à m’aider avec des suggestions et des conseils fructueux surtout dans la compréhension et la gestion de la base de données d’Acquisio.

RÉSUMÉ

Avec l'essor de l'Internet et des moteurs de recherche, le Web marketing est devenu un métier à part entière qui s'est développé de manière totalement disruptive par rapport au marketing classique. Plusieurs paramètres forment la clé de voûte de la réussite d'une campagne de Web marketing dont : le choix des mots clés, le budget à allouer pour chaque mot clé, une bonne appréciation des attributs des cibles (langue, localisation, …) etc. Ce paramétrage est une tâche complexe à cause des quantités gigantesques de données à traiter d'où le recours grandissant à des firmes spécialisées dans la gestion des campagnes publicitaires sur Internet.

Pour faire face à la très forte concurrence dans ce secteur de pointe, la société Aquisio, un leader mondial dans ce domaine, a lancé ce projet en collaboration avec l'École Polytechnique de Montréal. L'objectif étant de développer un module d'optimisation robuste permettant la gestion des campagnes publicitaires dans les moteurs de recherches d'Internet.

Le présent projet de maîtrise porte sur le développement et l'implantation d'un module d'optimisation, à base de la recherche Tabou, dont le but est de maximiser le rendement de toute une campagne publicitaire sur Internet. L'efficacité de notre approche de résolution a été prouvée par des tests réalisés sur des échantillons de six bases de données fournies par notre partenaire industriel. En effet, l'utilisation du Tabou nous a permis de nous affranchir des limitations des méthodes d'optimisation généralement implantées dans les outils commerciaux. De plus la solution finale générée par notre module atteint aisément les 95% de la solution optimale et ceci est vrai pour les trois solutions initiales testées. Outre la bonne qualité des solutions, nous nous sommes également intéressés au temps d'exécution. Ainsi, en réduisant la taille du voisinage, nous avons réussi à générer des solutions de bonne qualité en un temps de calcul raisonnable, de l'ordre de quelques minutes.

Mots clés : Web marketing, Optimisation, Recherche Tabou.

ABSTRACT

With the rise of the Internet and search engines, Web marketing has become a profession in its own that has disrupted the evolution of traditional marketing. Several parameters are key to a successful Web marketing campaign including the selection of keywords, the budget allocated for each keyword, a good understanding of the attributes of the target (language, location, etc.). This configuration is an increasingly complex task due to the gigantic quantity of data to be processed. Thus, the use of specialized firms' services for managing advertising campaigns on the Internet is continuously growing.

To cope with the strong competition in this leading sector, the company Aquisio, a world leader in this field, launched this project in collaboration with the Polytechnic School of Montreal. The objective is to develop a robust optimization module for managing advertising campaigns in Internet search engines.

This Master project focuses on the development and implementation of an optimization module, based on the Tabu Search, whose goal is to maximize the performance of any advertising campaign on the Internet. The effectiveness of our approach resolution was proved by tests conducted on samples of six databases provided by our industrial partner. Indeed, the use of Tabu has allowed us to overcome the limitations of optimization methods generally provided in commercial tools. In addition, the final solution generated by our module reaches easily 95% of the optimal solution and this is true for the initial three solutions tested. Besides the high quality of solutions, we are also interested to execution time. Thus, by reducing the size of the neighborhood, we were able to generate good solution quality in a reasonable computation time, on the order of a few minutes.

Key words: Web marketing, Optimization, Tabu Search

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Internet a connu une évolution phénoménale lors des deux dernières décennies aussi bien d'un point de vue technique que par son ubiquité et son adoption par le grand public. Il est tout naturellement devenu un média incontournable pour la promotion publicitaire de la plupart des produits commerciaux au même titre que la radio et la télévision avant lui. La publicité sur ce support a par ailleurs suivi la même courbe d'évolution rapide, dopée par l'innovation technique et les changements sociaux induits par l'appropriation de l'Internet par les utilisateurs. La publicité sur Internet aujourd'hui est radicalement différente par rapport à il y a une dizaine d'années. De nouvelles stratégies de ciblage publicitaire via la toile font surface et changent totalement la manière dont les agences de marketing transmettent leurs messages. La gestion d'une campagne publicitaire sur le Web devient de plus en plus complexe compte tenu de la quantité grandissante de données à considérer. En effet, une seule campagne peut comporter plusieurs milliers de mots clés, rendant nécessaire sa gestion par des professionnels. On assiste ainsi à l'émergence d'entreprises spécialisées dans la gestion des campagnes publicitaires sur Internet. Ces entreprises ont recours à des méthodes et algorithmes d'optimisation afin de maximiser leurs revenus tout en respectant certaines contraintes.

L'optimisation de la gestion de ces campagnes publicitaires sur Internet est un axe de recherche assez récent. Peu d'articles et de publications scientifiques traitent ce sujet. Ainsi, l'objectif de ce travail est de développer un module d'optimisation permettant la gestion efficace des campagnes publicitaires dans les moteurs de recherches d'Internet. Ce projet a été lancé en partenariat avec l'entreprise Aquisio un leader mondial dans le domaine du Web marketing.

L'approche proposée dans ce travail consiste à adopter la recherche Tabou qui a prouvé son efficacité dans la résolution de problèmes d'optimisation dans plusieurs autres domaines. L'utilisation du Tabou permet de s'affranchir des limitations des méthodes d'optimisation généralement implantées dans les outils commerciaux. Ainsi il sera possible de bénéficier de l'insensibilité du module d'optimisation à développer face aux problèmes à résoudre quels que soient leurs types, linéaires ou quadratiques.

Ce mémoire se divise en cinq chapitres. Au chapitre 1, on introduit et définit les notions de bases du Web marketing. Ensuite le chapitre 2 traite deux parties. La première partie présente une revue des travaux de recherche sur le Web marketing. La seconde partie expose l'algorithme

Tabou. Le chapitre 3 décrit la modélisation mathématique du problème. Le chapitre 4 est consacré à l'implémentation et l'amélioration de l'algorithme Tabou. Finalement, le dernier chapitre présente et analyse les résultats des tests numériques qui permettent de mesurer la performance du module d'optimisation proposé dans ce projet de maîtrise.

CHAPITRE 1 : MISE EN CONTEXTE

Internet est devenu depuis quelques années l'un des termes les plus utilisés dans le monde. Malgré son utilisation fréquente, la définition de ce terme du 20ième siècle est encore sujette à discussion.

Internet peut être considéré aujourd'hui comme un média, un espace social, une communauté, une bibliothèque gigantesque, …, une fenêtre sur le monde.

Selon l'encyclopédie Larousse [1], Internet est un « Réseau télématique international, qui résulte de l'interconnexion des ordinateurs du monde entier utilisant un protocole commun d'échanges de données (baptisé TCP/IP ou *Transport Control Protocol/Internet Protocol* et spécifié par l'*Internet Society*, ou ISOC) afin de dialoguer entre eux via les lignes de télécommunication (lignes téléphoniques, liaisons numériques, câble). »

L'Unité Régionale de Formation à l'Information Scientifique et Technique (URFIST) de Bretagne et des Pays de la Loire [2] s'est penchée à son tour sur la question en posant la question « Internet est un phénomène difficile à cerner et à définir avec précision : est-ce un espace, un média, un outil, une infrastructure, une culture…? ».

Cette question donne un assez bon aperçu de l'ampleur de ce phénomène jeune dont les premières versions ont vu le jour vers la fin des années 1950. D'après Statistique Canada [3], en 2010, 80 % des ménages canadiens ont accès au Web et passent en moyenne 44 heures par mois à naviguer sur la toile.

Enfin, le monde d'aujourd'hui est de plus en plus gourmand en Internet et le temps passé à naviguer sur le Web ne cesse de croître. Alors, il est évident que tous ces indicateurs attirent de plus en plus les entreprises afin de promouvoir leurs produits et services sur ce nouveau canal publicitaire.

Cette nouvelle chaîne de diffusion publicitaire donne naissance à tout un secteur économique qui est la publicité sur Internet communément appelé le Web marketing.

1.1 Mise en évidence de l'importance de la publicité Web

Depuis l'apparition des bannières publicitaires, première forme de publicité sur Internet, la publicité Web n'a cessé de gagner de l'ampleur dans le monde marketing. En effet, d'après le « IAB/PwC Digital Ad Revenue Report 2012 » [4] les revenus de la publicité sur Internet aux États-Unis ont enregistré une croissance de 5% durant la deuxième moitié de l'année 2012 par rapport à sa première moitié pour atteindre 8,7 milliards de dollars. Des revenus à la hausse de 14% ont été enregistrés aussi par rapport à l'année 2011.

Internet devient, de plus en plus, un média incontournable pour la diffusion publicitaire de la plupart des produits commerciaux. On assiste aujourd'hui à l'explosion d'une nouvelle manière d'acheter et de vendre qui se fait entièrement via la toile. De la promotion du produit ou service, en passant par la phase de commande et enfin le paiement, tout se fait entièrement par Internet. La publicité d'aujourd'hui n'est plus aussi basique que lors des premiers jours, des nouvelles stratégies de ciblage publicitaire font surface et changent totalement la manière dont les agences de marketing transmettent leurs messages. Grâce à l'évolution de la technologie, on est capable aujourd'hui de localiser la diffusion de la publicité. On arrive même, dans certains cas, à cibler une tranche d'âge ou à faire une sélection par sexe. Tous ces avantages qu'offre en exclusivité la publicité sur Internet ne font qu'accroître l'intérêt que portent les annonceurs à ce nouveau type de média [5].

1.2 Les différents types de publicité sur Internet

Plusieurs définitions de la publicité sur Internet ont été proposées. Yoo [6] la définit comme une forme de communication payante à un commanditaire à travers Internet dans le but de convaincre de l'offre. En ce qui concerne les différents types de publicité qui existent sur Internet, il faut savoir qu'il n'y a pas vraiment de classification universelle bien définie. Ceci dit, *l'Interactive Advertising Bureau* (IBA), propose une classification en neuf types de publicité sur Internet qu'on pourrait résumer par : *Search, Display / Banner, Classifieds, Digital Video, Lead Generation, Mobile, Rich Media, Sponsorship* et enfin courriel. Le rapport annuel de la publicité sur Internet d'IBA pour l'année 2011 [7] révèle que la catégorie Search est celle qui génère le plus de chiffre d'affaire comme l'indique la figure 1-1.

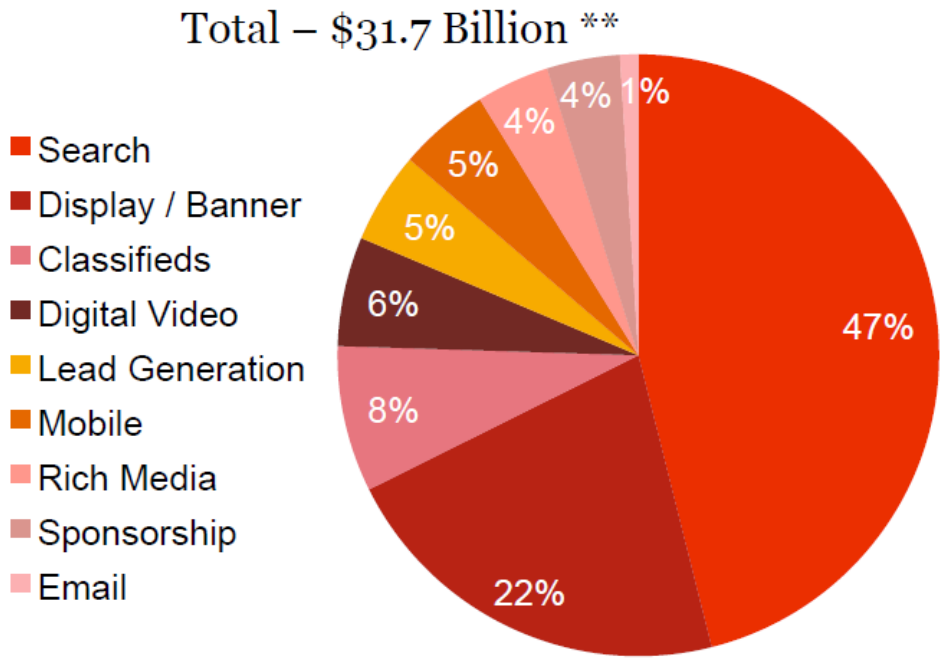

Figure 1-1: Répartition du chiffre d'affaires de la publicité sur Internet en 2011 : source [7]

La catégorie *Search* est celle qui sera retenue pour l'ensemble de ce mémoire. La publicité Web sur le réseau *Search* est celle qui s'affiche à l'écran de l'internaute suite à une requête ou une recherche lancée depuis un moteur de recherche. Elle est constituée de 2 à 4 lignes de texte décrivant le produit ou le service de l'annonce, et un lien pointant sur le site Web de l'annonceur (ou l'une de ses pages). Ce type de publicité Web, plus communément appelé annonce textuelle, se trouve généralement sur les pages des résultats de recherche. Tout un mécanisme complexe de système d'enchères est à l'origine de la classification des annonces. Ce mécanisme sera décrit plus en détails à la section 1.3.2. L'ordre d'apparition de ces annonces joue aussi un rôle très important dans l'efficacité des campagnes publicitaires. Ceci sera expliqué plus en détail à la section 1.3.3.

1.3 Mécanisme de la publicité Web « Search »

Cette section débute par la présentation de quelques définitions nécessaires pour la compréhension du fonctionnement des moteurs de recherches. On présentera ensuite, l'impact de la position des annonces sur la campagne publicitaire.

1.3.1 Définitions importantes

Ce paragraphe présente quelques définitions, exemples et explications propres au jargon du Web marketing et de la publicité Web. Ces explications seront d'une grande utilité pour la compréhension du reste de ce mémoire. Il est important aussi de mentionner que les définitions présentées dans la suite de cette section sont équivalentes et appropriées pour les cas de Google, Yahoo et Bing qui sont les trois plus grands moteurs de recherche.

Les différents termes à définir et à expliquer seront présentés dans un ordre alphabétique afin de faciliter la consultation tout au long de la lecture de ce mémoire.

- **Annonce textuelle « text ad »**

C'est la forme de publicité Web la plus utilisée sur le réseau Search. Elle a pour but d'attirer l'attention de l'internaute et l'amener à cliquer sur l'annonce qu'elle offre. Elle est constituée essentiellement de 2 à 4 lignes. La première ligne contient généralement le nom du produit ou service qui est à offrir (le titre). Quant à la deuxième ligne elle se présente sous forme d'une adresse URL qui pointe soit sur la page d'accueil du site Web de l'annonceur, soit sur une page bien précise choisie par ce dernier. Enfin, la troisième et éventuellement la quatrième ligne contiennent une brève description de l'offre.

Figure 1-2 : Exemple d'annonce textuelle

- **Annonceur « advertiser »**

Il peut prendre la forme d'une personne ou d'une entreprise qui cherche à promouvoir un produit ou service. L'annonceur a le choix de paramétrer et de gérer ses annonces tout seul, directement sur les plateformes de publicité Web des moteurs de recherche. Il peut aussi recourir aux services d'agences professionnelles de Web marketing pour le conseiller et gérer sa campagne publicitaire sur Internet.

❖ **Budget quotidien « daily budget »**

C'est le budget journalier qu'alloue l'annonceur à sa campagne publicitaire. Ce budget rétrécit à chaque fois qu'un internaute clique sur le lien de l'annonceur. Dès que ce budget devient inférieur au coût par clic, la campagne publicitaire est suspendue jusqu'à la fin de la journée.

❖ **Campagne publicitaire « campaign »**

Une campagne publicitaire sur Internet n'est pas si différente des campagnes publicitaires classiques. Toutes les deux ont nécessairement une date de début et éventuellement une date de fin, une localisation et des cibles bien précises telles que l'âge, le sexe, etc. Par contre, la campagne publicitaire sur Internet se distingue par sa longue liste de mots clés (on parle de plusieurs milliers de mots clés dans certains cas).

❖ **Clic « click »**

Un clic est comptabilisé à chaque fois qu'un internaute clique sur une annonce. Suite à cette action, l'utilisateur se voit dirigé vers la page cible de l'annonce en question et le coût du clic, qui dépend de la position de l'annonce, sera retranché du budget quotidien associé à cette annonce.

❖ **Conversion « conversion »**

La conversion est l'action associée à convertir le visionnage d'une annonce publicitaire par un internaute, à son passage à l'action. Il est de la responsabilité de l'annonceur de définir clairement ce qu'est une conversion. Elle peut être sous forme d'un achat en ligne, une prise de rendez-vous, la réponse à un questionnaire, ou encore simplement la visite d'une page précise du site Web de l'annonceur.

❖ **Coût par clic « cost per click (CPC) »**

C'est le coût facturé par l'entreprise propriétaire du moteur de recherche à chaque fois qu'un internaute clique sur l'annonce promue. Un algorithme calcule ce coût et fait par la suite payer à l'annonceur le prix le plus bas, pourvu que celui-ci soit supérieur à l'enchère mise par ses concurrents. Ceci implique que les coûts peuvent varier tout au long de la journée. C'est l'une des raisons pour lesquelles les moteurs de recherche sont incapables de produire des statistiques instantanées relatives au coût par clic. Une autre raison tout aussi importante qui pousse les

entreprises propriétaires des moteurs de recherche à ne pas donner le CPC exact avant l'enchère, et que ces entreprises ne veulent pas dévoiler leurs algorithmes.

- **Groupe d'annonces « ad group »**

Un groupe d'annonces est un sous-ensemble de la campagne publicitaire. De ce fait, une annonce peut contenir plusieurs groupes d'annonces distincts, dans le but d'augmenter l'efficacité de toute la campagne. En effet, cette structuration de la campagne publicitaire permet d'une part d'associer des créatifs (annonces textuelles) à chaque groupe d'annonces, et il est d'autre part ainsi plus facile de regrouper les statistiques.

- **Impression « impression »**

Une impression se produit lorsqu'une annonce textuelle relative à l'un des mots clés du groupe d'annonce s'affiche dans la page des résultats de la recherche lancée par un internaute sur un moteur de recherche. Cette impression sera comptabilisée pour le mot clé qui a été à l'origine de cette action.

- **Indice de qualité « quality score »**

C'est une mesure qui est calculée par les moteurs de recherche afin de pouvoir décider si une annonce quelconque est pertinente ou non par rapport à la requête qui est lancée par l'utilisateur. La manière dont la valeur de l'indice de qualité est calculée diffère d'un moteur de recherche à un autre et demeure un secret industriel. Ce qui est connu concernant la détermination de l'indice de qualité est qu'il est calculé à partir du taux de clics, de la pertinence entre le mot clé et la page de destination, et enfin la pertinence existante entre le mot clé et l'annonce en question.

- **Mot clé « keyword »**

Un mot clé n'est pas nécessairement constitué uniquement d'un seul mot. Plusieurs mots peuvent être assemblés afin de créer un mot clé. Partant de ce principe, on peut dire qu'il y a une infinité de combinaisons de mots, et donc une infinité de mots clés. Certes, ils ne sont pas tous utilisés, mais ceux qui le sont doivent avoir une valeur d'enchère fixée par l'annonceur. Plus cette valeur d'enchère est grande, plus l'annonce textuelle de l'annonceur aura des chances d'apparaître dans les premières positions, dépendamment des compétiteurs et de la pertinence du mot clé.

❖ **Moteur de recherche « search engine »**

D'après Web-marketing, « Le moteur de recherche est un outil de recherche qui référence automatiquement les pages Web se trouvant sur le réseau à l'aide d'un programme appelé spider ou robot ou après soumission de l'URL par un responsable du site. Le contenu de chaque page est analysé et les pages sont classées par ordre de pertinence dans les pages de résultats en fonction des mots clés saisis par les utilisateurs lors de leurs requêtes sur le moteur de recherche » [8]. Ce classement s'effectue en fonction de l'indice de qualité défini plus haut.

❖ **Position « position »**

La position peut être considérée comme le rang qu'attribuent les moteurs de recherche aux différentes annonces textuelles. Dépendamment des compétiteurs et de leurs CPC maximaux, plus une annonce est pertinente, plus son indice de qualité est grand, et plus la valeur de sa « position » est petite. Il y a aussi les positions dites « Premium » qui seront sélectionnées par les moteurs de recherche selon la pertinence des mots clés. Ce type de position est affiché directement au-dessus de la page des résultats de la requête comme indiqué à la figure 1-3. Sur cette figure, les numéros indiquent l'ordre des positions tel que géré par Google. Généralement, on ne trouve pas plus que trois positions Premium et dix annonces publicitaires dans la page de résultats des moteurs de recherche.

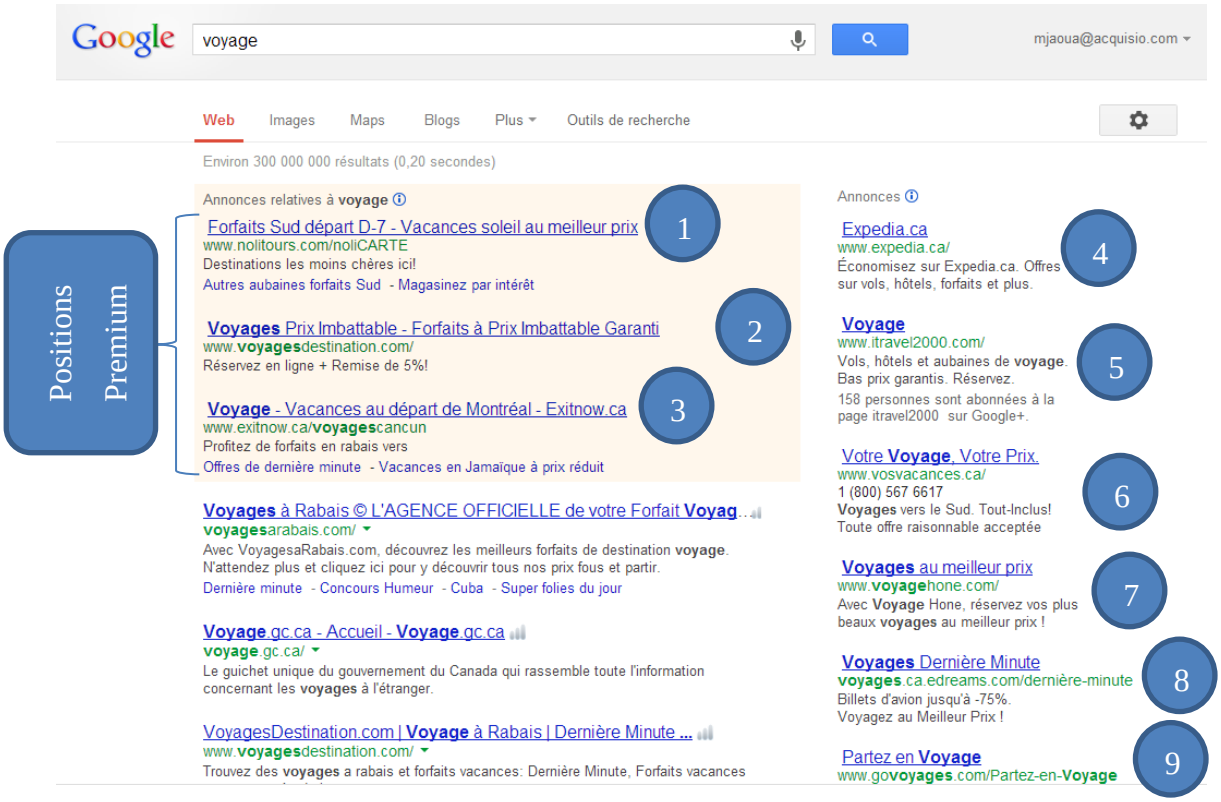

Figure 1-3 : Exemple de la page de résultats de Google

❖ Client potentiel « prospect »

Un client potentiel ou encore *prospect* en anglais n'est pas tout à fait un client puisqu'il n'a encore rien acheté. Cela dit, ce qui caractérise le prospect est qu'il a de fortes chances de devenir un client dans un futur proche. Tous les internautes qui sont sur le réseau *Search* et qui font partie du groupe ciblé d'une annonce publicitaire peuvent être considérés comme des clients potentiels. Dès qu'ils passent une commande ou dès qu'une conversion se produit (dans le cas du Web marketing), ils deviennent des clients.

❖ Requête « query »

Une requête, dans le sens large du mot, désigne une demande. Dans le cas des moteurs de recherche, une requête est aussi une demande, mais sous forme d'une expression formatée selon des règles définies par les concepteurs du moteur de recherche en question. Elle a pour objectif de sélectionner, de trier et d'ordonner les sites Web selon leur pertinence.

- **Type de correspondance « match type »**

Dans les moteurs de recherche actuels, le type de correspondance est un paramètre très important lors de la sélection des résultats de la recherche lancée. La détermination de ce paramètre diffère légèrement d'un moteur de recherche à un autre, mais généralement le *match type* donne une idée précise du degré de similarité existant entre la requête et le mot clé. On présente dans la liste suivante les plus importants types de correspondances qui sont mis à la disposition des annonceurs :

- Expression exacte

  Quand ce type de correspondance est demandé, tous les mots clés figurants dans l'expression doivent absolument se présenter dans l'ordre dans le contenu des annonces sélectionnées par le moteur de recherche. La présence d'autres mots qui succèdent ou précédent l'expression exacte recherchée n'est d'aucune influence sur le processus de sélection,

- Expression négative

  Lors de l'activation de ce paramètre, toutes les annonces sélectionnées ne doivent pas contenir cette expression négative.

- Mot clé exact

  Ce paramètre impose que toutes les annonces sélectionnées par la requête contiennent exactement le mot clé demandé.

- Requête large

  Ce type de requête permet au moteur de recherche de sélectionner les annonces qui contiennent les mots clés spécifiés peu importe l'ordre. Il autorise aussi la présence d'une légère modification de type syntaxique (singulier/pluriel) ou sémantique (synonyme).

- **Utilisateur « user »**

Le terme utilisateur est employé dans ce mémoire pour désigner l'internaute qui lance une requête sur le moteur de recherche.

❖ **Valeur d'enchère « bid » ou « CPC max »**

C'est le prix maximum que l'annonceur est prêt à payer pour un clic. Un mécanisme d'enchère est derrière l'ordonnancement des annonces textuelles qui figureront dans la page des résultats. Actuellement, ce sont des variantes de GSP *« Generalized Second-Price »* qui sont utilisées, mais historiquement ce n'était pas le cas. Ce mécanisme d'enchère peut être résumé comme suit: l'algorithme *Generalized Second-Price* va sélectionner les annonces dans un ordre décroissant de pertinence. Ce classement se fera en calculant le produit de la valeur de l'enchère avec l'indice de qualité de chaque annonce. Suite à cela, l'algorithme en question affichera les différentes annonces selon l'ordre qu'il aura établi (la position 1 est celle qui est placée tout en haut de la liste des résultats, comme indiqué à la figure 1-3). Nous précisons que la valeur de l'enchère qui est le coût fixé par l'annonceur pour un clic sur un mot clé donné ne sera jamais dépassée par ce mécanisme.

1.3.2 Fonctionnement de la publicité dans les moteurs de recherche

Les moteurs de recherche, comme définis dans la section 3.1 sont des algorithmes qui interagissent avec le réseau le plus étendu au monde qu'est Internet. L'utilisateur introduit dans ce qu'on appelle la barre de recherche les mots clés qui l'intéressent. Lorsque cette recherche ou requête est lancée, les algorithmes de recherche sélectionnent les sites et pages Web qui sont en relation avec les termes inscrits par l'internaute. La sélection des liens les plus pertinents se fait selon plusieurs critères. Tout l'art des moteurs de recherche performants repose sur ces critères. Ainsi, il est évident que les compagnies de moteurs de recherche telles que Google et Bing ne divulguent pas leurs secrets. Ceci dit, il est clair que cette sélection repose sur l'indice de qualité, le type de correspondance, le nombre d'occurrences du mot clé dans le site et la présence des mots clés dans la liste des métadonnées (une liste de mots clés en rapport avec le site Web mais qui sont invisibles à l'utilisateur).

Pour les annonces textuelles, Google a développé en 2002 un algorithme portant le nom de GSP « *Generalized Second-Price* ». Cet algorithme a la responsabilité de trier les annonces. Bien évidemment, depuis 2002, GSP a eu l'occasion d'être amélioré plusieurs fois. Mais son

fonctionnement de base, qui sera brièvement expliqué dans le paragraphe suivant, est resté le même [9].

Le choix des annonces textuelles qui seront sélectionnées sur la page des résultats s'effectue de la même façon ou presque que la recherche habituelle dans les moteurs de recherches. La différence majeure est la prise en compte du coût par clic (CPC) dans l'ordonnancement de ces annonces. Chaque annonceur doit fixer un CPC maximal pour chacun des mots clés inclus dans son groupe d'annonces. Ensuite, lors du lancement d'une requête sur un moteur de recherche, celui-ci sélectionne toutes les annonces ayant misé sur le mot clé figurant dans cette requête. Suite à cela, le moteur de recherche ordonne toutes les annonces sélectionnées dans un ordre décroissant du produit de la valeur d'enchère et de l'indice de qualité. Ainsi, à chaque fois que l'une de ces annonces est cliquée, le CPC que devra payer son annonceur est égal au prix minimal qui permet à l'annonce de garder sa position dans la liste établie par le moteur de recherche.

1.3.3 Importance de la position de l'annonce

Si les moteurs de recherche tirent un grand profit du positionnement des annonces textuelles, c'est qu'il y a bien une raison derrière cela. En effet, des études ont démontré [10] l'impact que joue la position de l'annonce textuelle sur le rendement de l'annonce publicitaire. Ceci implique que les annonceurs vont devoir payer le prix le plus élevé afin de voir leurs annonces dans les premières positions des pages de résultats.

Le nombre d'articles scientifiques qui mettent en évidence la relation de cause à effet entre la position d'une annonce textuelle et la probabilité qu'elle soit cliquée est assez limité. Selon nos recherches, le seul travail de recherche pertinent qui a traité ce sujet a été mené par Agarwal, Hosanagar et Smith en 2008 [11]. Mais il suffit d'avoir un peu de bon sens et un bon raisonnement logique pour tirer la conclusion que les premières annonces textuelles présentées sont plus probables d'être cliquées. L'étude faite par Agarwal et al. [11] confirme bien cette déclaration. Les auteurs de cette étude attestent que le nombre de clics (taux de clic) sur une annonce textuelle est fortement influençable par son positionnement sur la page des résultats de la recherche. Enfin, ils concluent que dans la grande majorité des cas, la courbe des taux de clics d'un mot clé est décroissante en fonction de la position que prend l'annonce textuelle. Enfin il est important de mentionner que tout le travail réalisé au cours de ce projet reposera sur ce postulat.

1.4 Gestion de campagne publicitaire sur Internet

Comme pour toute campagne publicitaire, peu importe le type de média, les campagnes publicitaires sur Internet doivent leurs réussites à une bonne gestion. De nos jours, il est loin d'être suffisant de ne lancer qu'une action marketing pour pouvoir récolter ses fruits. Particulièrement dans le cas des campagnes marketing sur Internet, leur gestion demande une assez bonne connaissance du domaine. Le choix des mots clés, du budget à allouer pour chacun d'eux, des cibles (langue, localisation, etc) et plusieurs autres paramètres forment les clés du succès de toute campagne publicitaire sur le Web. C'est pour cette raison que des entreprises spécialisées dans la gestion des campagnes publicitaires sur Internet voient de plus en plus le jour. Dans la prochaine sous-section, on présente les objectifs visés par les campagnes publicitaires sur Internet. Ensuite, le rôle des algorithmes d'optimisation sera explicité tout en faisant référence à l'importance que joue ce type d'algorithme dans la gestion des campagnes publicitaires sur Internet.

1.4.1 Objectifs

Toute entreprise qui décide de se lancer et d'investir dans un plan marketing, le fait avec l'espoir d'augmenter ses ventes. De la même manière, dans le cas des campagnes promotionnelles sur Internet, les entreprises espèrent voir leurs chiffres d'affaires augmenter. Toute la problématique qui se pose à la publicité sur Internet, vient du fait qu'il n'est pas toujours évident de quantifier l'apport direct que génère la publicité Web. Cette difficulté naît du fait que l'indicateur de performance en lui-même n'est pas chose facile à définir. Le premier indicateur de performance qui vient à l'esprit est probablement le nombre de conversions. Mais en réalité il n'est pas toujours possible de récolter les statistiques relatives à cette mesure. Pour cette raison, un indicateur de performance qui serait plus global et générique serait le taux de clics sur les annonces textuelles. Afin de pouvoir quantifier ce taux, il est impératif de connaitre le nombre d'impressions de cette annonce textuelle et le coût qui sera généré. Partant de cette affirmation, les indicateurs de performances qui seront pris en considération tout au long de cette étude sont :

- Le nombre de clics qu'on cherche à maximiser.
- le nombre de coût par clic qu'on cherche à maximiser, tout en augmentant le nombre de clics.

- Le nombre d'impressions qui est un critère de qualité minimale à respecter.
- Le budget qui est une contrainte critique et selon le choix de l'annonceur ce budget doit être dans certains cas totalement épuisé et dans d'autres il ne faudra juste pas le dépasser.

Il serait possible aussi d'introduire d'autres objectifs ainsi que d'autres critères de qualité.

Ce qui intéresse réellement les annonceurs est d'avoir un maximum de retour sur leurs investissements dans la publicité. Donc, toute l'expertise des agences de Web marketing repose sur la bonne gestion des campagnes publicitaires de leurs clients. En effet, d'une part ils doivent respecter le budget investi dans la publicité, et d'autre part, ils ont la responsabilité d'augmenter le nombre de clics et le nombre d'impressions. Sachant qu'un seul groupe d'annonces peut comporter plusieurs milliers de mots clés, leur gestion n'est pas une mince affaire. C'est là où intervient l'outil automatique du traitement de l'information, qui a la capacité de traiter une grande quantité d'information en peu de temps.

1.4.2 Rôle des algorithmes d'optimisation

Il est certain qu'avec une bonne connaissance du domaine, un annonceur pourrait gérer lui-même une campagne publicitaire sur Internet. Toutefois, vu la complexité de la tâche du point de vue de la quantité de données, il ne sera jamais capable de l'optimiser au même niveau que les algorithmes d'optimisation. C'est ce qui explique la concurrence féroce qui s'installe entre les entreprises de gestion des campagnes publicitaire à travers le monde. Les annonceurs exigent aujourd'hui la perfection dans le but de maximiser le retour sur l'investissement effectué pour la publicité.

Grâce à un bon algorithme d'optimisation des campagnes publicitaires sur Internet, les annonceurs pourront gagner énormément de temps tout en ayant de meilleurs résultats. L'algorithme en question aura comme tâche de maximiser le profit (nombre de clics), tout en s'assurant d'avoir un pourcentage d'impression au-dessus d'une borne minimale et tout en respectant le budget alloué pour la campagne.

C'est suite à ce besoin, devenu depuis quelques années indispensable aux agences spécialisées dans le marketing sur Internet, que ce projet de maîtrise a vu le jour. En effet, dans le cadre d'un partenariat entre le département de mathématiques et de génie industriel (MAGI) de l'École Polytechnique de Montréal et l'entreprise Acquisio spécialisée dans la gestion des

campagnes marketing sur le Web, ce projet de maîtrise vise à développer un algorithme performant, permettant d'optimiser la gestion des campagnes publicitaires sur Internet.

1.5 Présentation d'Acquisio

Acquisio est un leader mondial dans le domaine de la gestion des campagnes publicitaires sur Internet. Cette entreprise offre aux annonceurs une plateforme logicielle facilitant la gestion de leurs campagnes. On rapporte dans ce paragraphe, la présentation d'Acquisio telle qu'exposée dans leur site Web : www.acquisio.com.

« À titre de chef de file mondial des plateformes de solutions médias liées au rendement, Acquisio aide les spécialistes du marketing à acheter, à effectuer le suivi, à gérer, à optimiser et à produire des rapports portant sur les placements médias dans tous les réseaux. La plateforme a été conçue pour la gestion de campagnes dans les moteurs de recherche, mais elle a évolué pour inclure la publicité dans Facebook et dans les principaux réseaux d'affichage de publicités en temps réel. Acquisio met de l'avant une technologie à la fine pointe de sa catégorie au profit des spécialistes du marketing qui achètent des placements médias dans n'importe quel réseau en ligne, leur permettant d'accomplir toutes les tâches associées à la publicité liée au rendement, de l'achat des placements jusqu'au suivi des conversions et bien plus encore, à même une seule plateforme intégrée.

À titre d'outil de gestion de plus de 500 millions de dollars publicitaires, Acquisio est la solution publicitaire multiréseau privilégiée par les annonceurs et les agences de marketing. Les solutions que propose Acquisio autonomisent plus de 300 agences interactives à l'échelle mondiale, entre autres, Bruce Clay, Groupe Pages Jaunes, SEO Inc., aimClear, Page Zero, Find Me Faster, Leverage Marketing et SEER Interactive, les aidant à accroître la productivité et l'efficacité de leur personnel, ainsi que le RCI « Retour sur Capitaux Investis » qu'elles proposent à leurs clients » [12].

1.6 Présentation du sujet de recherche

Le but principal de ce mémoire est de développer un module d'optimisation pour les campagnes publicitaires sur Internet. Pour ce faire, une bonne implémentation d'une métaheuristique sera capable de répondre aux besoins.

Dans un premier temps, le problème à résoudre pourrait être vu comme le problème du sac à dos qui est un modèle assez connu par les mathématiciens. La formulation la plus connue du problème du sac à dos consiste à sélectionner à partir d'un ensemble de boîtes ayant chacune un poids connu et une valeur donnée, les boîtes qui maximisent le contenu total du sac sans dépasser sa limite de poids. Vu la complexité et la grande dimension de l'espace de recherche, une approche de résolution par une méthode exacte serait inappropriée. Ceci est dû au facteur temps de réponse de l'algorithme à développer, qui est une contrainte assez importante. De plus, notre partenaire industriel impose certaines exigences : il ne veut pas que sa plateforme de résolution dépende d'un logiciel commercial ou même que le code source du module d'optimisation à développer soit tiré de l'un des code source des logiciels libres tel que GLPK (*GNU Linear Programming Kit*) [13]. Finalement, certains critères d'évaluation que les clients souhaiteraient optimiser ou satisfaire impliquent des équations non linéaires. Pour toutes ces raisons, nous optons pour une approche de résolution heuristique.

Une multitude de choix parmi les algorithmes métaheuristiques se présente. À commencer par les algorithmes génétiques présentés la première fois par John Henry Holland en 1975 [14] Ces algorithmes évolutionnaires se basent sur la théorie de la sélection naturelle énoncée par Charles Darwin en 1859 [15]. En effet, les algorithmes génétiques s'inspirent de la notion de l'évolution naturelle en appliquant ses principes pour la résolution des problèmes d'optimisation. Face à un problème d'optimisation présentant un très grand nombre de solutions, les algorithmes génétiques se servent des principes d'évolution naturelle et de sélection des espèces afin de résoudre ce genre de problème. Ces principes se présentent sous trois formes. Premièrement, on trouve la notion de croisement entre individus (ou solutions) afin d'en générer des nouveaux. Deuxièmement, on assiste de temps à autres à des mutations qui apparaissent aléatoirement au sein d'un ensemble de solutions. Troisièmement, on trouve la notion de sélection des individus ou des solutions qui est d'une grande importance puisqu'elle permet de retenir les solutions les plus prometteuses. Ainsi, en se basant sur ces phénomènes inspirés de la nature, les algorithmes génétiques recombinent des solutions pour en générer de nouvelles. Et grâce au processus de sélection, seule les bonnes solutions seront gardées.

Nous envisageons ensuite un autre algorithme faisant partie de la famille des métaheuristiques appelé le recuit simulé. Cette métaheuristique s'inspire d'un processus physique utilisé en métallurgie dans le but d'améliorer la qualité d'un solide. Au cours des années 1950,

Metropolis-Hastings et al. [16] réalisent des expériences afin de simuler les étapes d'évolution d'un système thermodynamique. Metropolis et al. [16] parviennent à minimiser l'énergie d'un matériau en alternant des cycles de refroidissement et de réchauffage, et ce processus leur permet en fin de compte d'améliorer la qualité d'un solide. S'inspirant de ce processus physique, Kirkpatrick et al. [17] mettent au point une nouvelle méthode de recherche qu'ils nomment le recuit simulé. Par analogie au processus physique où l'on minimise l'énergie du système, l'algorithme du recuit simulé tend à minimiser une fonction généralement appelée E. Cet algorithme se sert aussi d'un paramètre fictif nommé T par référence à la température du système. D'une itération à une autre, l'algorithme du recuit simulé essaiera de minimiser la fonction E tout en jouant sur la valeur du paramètre de température T. Plus la valeur de T est grande plus l'algorithme est libre d'exploiter l'espace de recherche. Par contre, plus la température est basse, plus l'algorithme va intensifier la recherche autour d'un point donné. Ainsi, l'algorithme du recuit simulé tentera d'atteindre l'optimum en mettant à jour durant chaque itération la valeur de T afin d'orienter la stratégie de la recherche.

Hormis les deux exemples de métaheuristiques décrits précédemment, un grand choix d'algorithmes de la même famille nous est offert. Mais finalement nous optons pour une tierce méthode métaheuristique baptisée la recherche Tabou. Dans la littérature, nous avons constaté que l'utilisation de cette méthode dans le domaine de la finance est avantageuse. En effet, des travaux menés par Majid Aldaihani et al. [18] ont démontré l'efficacité de l'algorithme Tabou dans divers types de problèmes dans le secteur de la finance et plus spécifiquement dans l'optimisation des portefeuilles. Oswald Marinoni et al. [19] utilisent aussi l'algorithme Tabou pour l'optimisation du portefeuille d'investissements. On trouve encore Franco Rex Busetti [20] qui a travaillé sur l'optimisation des portefeuilles par des méthodes métaheuristiques dont le Tabou. Étant donné que notre problème pourrait être vu comme étant un problème de gestion des portefeuilles pour la publicité sur Internet, une certaine similarité est constatée entre notre problème et ceux étudiés par Majid Aldaihani et al. [18].

Ainsi ce travail aura comme objectif principal de développer et d'implémenter un module d'optimisation basé sur l'algorithme de la recherche Tabou. L'implémentation de cet algorithme permettra aux clients du partenaire industriel de déterminer la valeur minimale du CPC à affecter aux différents mots clés utilisés dans chaque campagne. Cette recherche s'effectuera tout en

optimisant un objectif propre au client et en respectant des critères de qualité qui se traduiront par des contraintes dans le modèle mathématique.

CHAPITRE 2 : REVUE DE LITTÉRATURE

Avant de commencer l'implémentation de l'algorithme Tabou pour l'optimisation du rendement des campagnes publicitaires, il est important de connaître l'état d'avancement de la recherche dans ce domaine. La publicité sur Internet est un domaine relativement jeune, le nombre de publications scientifiques traitant de l'optimisation de l'efficacité de ce type de campagnes publicitaires est de fait assez réduit. En outre, la concurrence très rude entre les entreprises de moteurs de recherche fait que les algorithmes d'optimisation développés par leur département de recherche et développement sont gardés jalousement secrets et confidentiels.

Dans cette revue de littérature, nous tenterons de couvrir trois grands volets. Le premier sera consacré à l'étude du fonctionnement de la publicité sur Internet. Le second jettera la lumière sur les courbes génériques de prédiction qui ont été développées par Patrick Quinn [21]. Finalement la troisième partie vise à expliquer le fonctionnement de l'algorithme de la recherche Tabou.

2.1 Les moteurs de recherche et le Web marketing

Avec la quantité d'information disponible aujourd'hui sur la toile, chercher une information sur Internet n'est plus possible sans l'aide d'un outil ou application Web référençant plusieurs milliards de sites et de pages Internet [22]. Cette infrastructure qui permet à l'internaute de trouver rapidement l'information qu'il désire est connue sous le nom de moteur de recherche. Depuis qu'Internet est devenu accessible au grand public, de nombreux moteurs de recherche ont vu le jour. Certains ont périclité, voire disparu, d'autres sont devenus des géants aux revenus gigantesques. Actuellement, on trouve aux premières places en termes de revenu généré, Google [23], Yahoo [24] et Baidu [25]. D'une part, les revenus générés à partir de ces moteurs de recherche s'élèvent à des milliards de dollars, et d'autre part lancer une recherche sur n'importe lequel de ces moteurs de recherche est totalement gratuite. Alors comment ces entreprises arrivent-elles à engendrer de telles sommes?

Le domaine du marketing interactif, plus communément connu sous le nom de Web marketing, est un domaine d'activité assez récent. En effet, ce secteur d'activité n'a vu le jour que durant les années 1990 en même temps que la globalisation s'est accélérée et que l'accès grand public à l'Internet s'est démocratisé.

Plusieurs variantes de la publicité sur Internet ont vu le jour. Les plus importantes sont les annonces textuelles sur les moteurs de recherches, les publicités par bannières et les publicités sur les réseaux sociaux. D'après Agarwal et al. (2008) [11] on trouve en tête de la liste la publicité sponsorisée, atteignant 40 % du marché des publicités interactives.

Jansen et Mullen [25] présentent une bonne introduction aux différents concepts du Web marketing et plus spécifiquement des publicités sur Internet de type annonce textuelle. Ils commencent par rappeler la nécessité des moteurs de recherche pour la navigation sur Internet. Suite à cela, les auteurs nous dévoilent la source de revenus de ces moteurs de recherche en nous expliquant la notion des enchères des mots clés. Par la suite, ils conceptualisent le processus de la recherche sponsorisée comme étant un aspect de la recherche d'informations sur Internet, tout en expliquant le fonctionnement et les mécanismes derrière ce type de recherche. Ils fournissent aussi un historique sur l'évolution des mécanismes de classement des annonces depuis les débuts de leur apparition en 1994 où la publicité sur Internet se limitait à la forme de bannières. En conclusion cet article forme une excellente introduction pour la compréhension du monde de la publicité sur Internet et des annonces textuelles.

Maintenant que le concept de la recherche sponsorisée est défini, passons à l'exploration et la compréhension de la stratégie employée par les moteurs de recherche dans le classement des annonces textuelles.

GSP *« Generalized Second Price »*, est actuellement le modèle le plus connu pour le classement des annonces textuelles. Ce modèle fut le sujet principal de l'étude menée par Edelman et al. dans leur article paru en 2006 [9]. Dans cet article, les auteurs nous détaillent le fonctionnement de l'algorithme GSP. Ce modèle qui émerge de l'évolution de plusieurs autres algorithmes de classement est adopté aujourd'hui par certains moteurs de recherche leaders tels que Google et Bing. Edelman et al. dévoilent dans leur article [9] tout le mécanisme tournant derrière GSP. En effet ils nous apprennent que GSP commence par calculer pour chaque annonce son score en multipliant la valeur de l'enchère proposée par l'annonceur, par l'indice de qualité. Il classe ensuite les annonces selon un ordre décroissant du score. Finalement, le prix qui sera facturé à l'annonceur, s'il y a un clic, sera égal au strict minimum pourvu que ce montant permette à l'annonce d'avoir un score meilleur que celui de l'annonce qui la suit.

2.2 Les courbes génériques de prédiction

La qualité de la solution qui sera proposée aux annonceurs ne dépend pas uniquement des performances du module d'optimisation. En effet, la qualité des données de départ ne manque pas d'importance. Il est vrai que le rôle de tout module d'optimisation est de trouver la meilleure solution à partir d'un ensemble de données de départ qui ont été fournis à l'optimiseur. Mais il est évident que si les données initiales manquent de précision, alors la solution fournie par n'importe quel optimiseur n'aura jamais un rendement optimal.

C'est là où intervient tout le travail de Patrick Quinn [21]. Partant de la constatation émise par Hou, Wang et Yang (2008) [10] qui révèle l'existence d'une forte interaction entre la position de l'annonce, la valeur de l'enchère, le nombre de clics et le nombre d'impressions, Quinn [21] tente de modéliser cette relation de causalité.

Dans son travail, Quinn [21] utilise la méthode de régression sur les données historiques de notre partenaire industriel pour formuler ses fonctions génériques de prédictions. Il commence par extraire des échantillons de différentes bases de données de notre partenaire industriel. Ensuite il sélectionne les mots clés qui ont un bon potentiel de prédiction. Une fois que la phase de sélection et de filtrage des données est terminée, Quinn exploite ces échantillons pour produire ses courbes génériques de prédiction pour le nombre de clics et pour le coût par clic. Il débute par une régression linéaire, mais n'étant pas satisfait de la qualité de ses résultats, il passe à la régression exponentielle grâce à laquelle il réussit à atteindre un résultat satisfaisant. Il propose deux fonctions de prédictions. La première est une fonction générique du nombre de clics en fonction de la position moyenne du mot clé $clics = k_{clics} * e^{m*pos}$. La deuxième décrit l'évolution du coût par clic moyen en fonction de la position moyenne des mots clés $CPC = k_{CPC} * e^{n*pos}$. Avec pos qui représente la position moyenne et les paramètres k_{clics} et k_{CPC} qui représentent les valeurs à l'origine de chacune des deux fonctions de prédiction. En ce qui concerne les paramètres *m* et *n*, ils décrivent l'allure décroissante des courbes exponentielles. On expose dans la figure 2.1 un exemple de la courbe de prédiction de la première fonction générique. La deuxième figure 2.2 présente, quant à elle, un autre exemple : celui de la courbe générique de CPC.

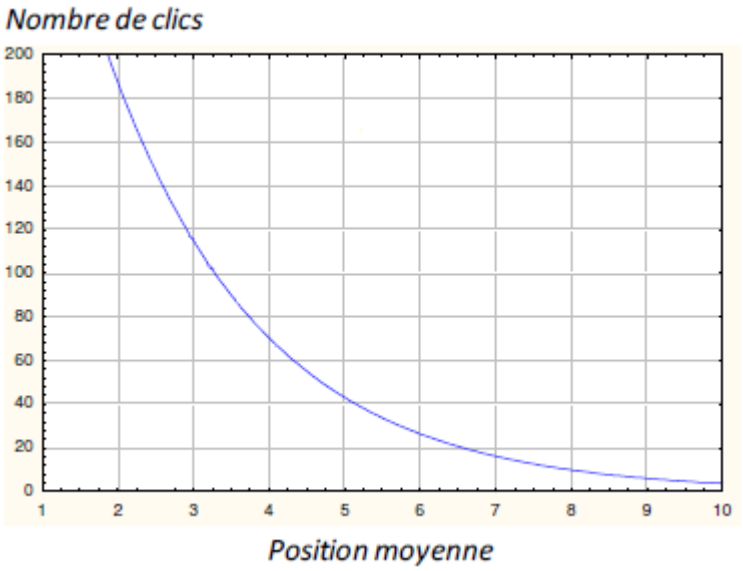

Figure 2-1: Exemple de la fonction de prédiction du nombre de clics : source [21]

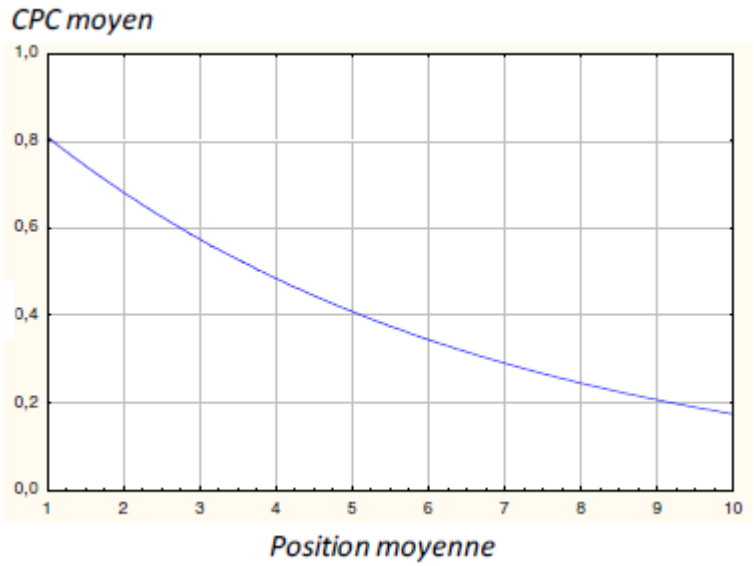

Figure 2-2 : Exemple de la fonction de prédiction du coût par clic moyen : source [21]

À partir de ces courbes génériques que Patrick Quinn a développées, nous sommes capables désormais de prédire pour n'importe quel mot clé et pour n'importe quelle position de 1 à 10, une estimation du nombre de clics, d'impressions et le coût par clics que le mot clé en question générera. C'est de cette manière que le programme basé sur l'algorithme Tabou procédera pour estimer la somme des clics, la somme des impressions et le coût total. En ce qui concerne l'estimation du coût, nous le calculons à partir du produit du CPC et du nombre de clics

moyen pour un mot clé donné et une position donnée. Ainsi, en se basant sur ces différents paramètres, le module d'optimisation sera capable d'évaluer et de comparer différentes solutions et d'en ressortir la meilleure.

2.3 La recherche Tabou

Dans certains problèmes d'optimisation, les méthodes de résolutions dites exactes, ne permettent pas trouver la solution optimale dans une durée de temps raisonnable.

C'est l'une des principales raisons qui ont contribué à la naissance des métaheuristiques. En effet cette famille d'algorithmes permet de résoudre des problèmes d'optimisations complexes face auxquels les méthodes classiques manquent d'efficacité. Cependant ces algorithmes de recherche ne peuvent garantir l'optimalité de la solution trouvée. Dans cette section, nous présentons la métaheuristique dont nous nous servirons pour optimiser la gestion des campagnes publicitaires.

En 1986 Fred Glover [26] dévoile pour la première fois sa métaheuristique qu'il baptisa la Recherche Tabou (RT). Cette appellation est tout à fait significative puisque cette méthode, se basant sur une recherche itérative qualifiée de recherche locale au sens large, interdit pour un laps de temps donné de revisiter une solution déjà visitée.

2.3.1 Algorithme de la recherche Tabou

La meilleure analogie sans doute pour expliquer l'idée derrière l'algorithme RT serait la fable des randonneurs [27]. Imaginons un randonneur malchanceux perdu dans la montagne. Il voudrait rejoindre le point de plus basse altitude puisqu'il sait qu'une équipe de secours passe régulièrement par ce point. Il ne sait pas qu'elle est l'altitude de son objectif et un brouillard l'empêche de voir loin. Face à un tel problème, la méthodologie de la recherche Tabou lui permettrait d'atteindre son objectif. Partant de n'importe quelle position du randonneur, la RT lui proposera de se déplacer vers le point qui le mènera à la plus basse altitude que le brouillard lui permette de voir. Une fois arrivé, il devra chercher de nouveau parmi toutes les positions qu'il peut apercevoir, la prochaine plus basse position pourvu qu'il ne l'ait pas déjà visitée. Il arrive dans certains cas que la position choisie soit de plus haute altitude que la position actuelle du randonneur. Le fait de remonter à cette nouvelle position, peut s'avérer une bonne décision puisque cette action peut mener le randonneur à de nouvelles positions encore plus basses que

celle qu'il a trouvé depuis son départ. En poursuivant ainsi de suite, le randonneur se déplacera d'un point à un autre tout en mémorisant les T dernières positions qui lui seront taboues.

Il est vrai que la fable des randonneurs [27] favorise la compréhension de l'algorithme Tabou. Mais dans le cadre d'un travail de recherche, il est impératif de passer à une description formelle et rigoureuse de cet algorithme. Dans cette optique, on s'est inspiré de la description d'Alain Hertz [29] de l'algorithme Tabou qu'on présente comme suit :

❖ Définition du problème :

Soit N l'ensemble de toutes les solutions possibles, et F une fonction à optimiser et qui détermine la valeur F(S) de toute solution S dans N.

Le problème à résoudre est donc le suivant :

$$\text{Max } F(S)$$

$$\text{Sous contraintes : } S \in N$$

On appelle voisinage, la fonction V qui associe un sous-ensemble de N à toute solution $S \in N$. Ainsi un voisin de S est toute solution $S' \in V(S)$.

Une solution $S \in N$ est considérée comme étant un maximum local dans un voisinage V si : $F(S') \leq F(S) \quad \forall S' \in V(S)$.

Une solution $S \in N$ est dite un maximum global si : $F(S') \leq F(S) \quad \forall S' \in N$

❖ Principe :

Durant l'évolution itérative de la recherche Tabou, cet algorithme choisit à chaque itération la meilleure solution $S' \in V(S)$, même si $F(S) > F(S')$. Quand la recherche atteint un maximum local S dans un voisinage V, l'algorithme Tabou sera contraint de se déplacer vers une solution S' avec $F(S) > F(S')$. Dans certain cas, il arrive que le voisinage de la solution S' contienne la solution $S \in V(S')$. Alors, même si la solution S est un maximum local dans le voisinage V, il ne faut surtout pas revenir immédiatement à S, sinon la recherche se trouverait piégée à tourner en rond entre S et S'. Afin d'éviter ce problème, l'algorithme Tabou fait intervenir une liste T baptisée liste taboue. Cette liste permet de mémoriser durant un nombre limité d'itérations (mémoire à court terme), les dernières solutions visitées et d'interdire tout

déplacement vers ces solutions. On appelle toute solution figurant dans la liste T une solution taboue.

❖ Algorithme de la recherche Tabou

Soit :

- S : Solution courante;
- S* : Meilleure solution rencontrée depuis le début de la recherche.

1. Choisir une solution $S \in N$, poser $T:=\emptyset$ et $S^*:=S$;
2. Tant qu'aucun critère d'arrêt n'est satisfait faire
3. Déterminer une solution S' qui maximise F(S') dans $V^T(S)$

   Avec $V^T(S) = \{ S' \in V(S)$ tel que $S' \notin T \}$
4. Si $F(S') > F(S^*)$ alors poser $S^*:=S'$
5. Poser $S:=S'$ et mettre à jour T
6. Fin du tant que

À part le fait que l'algorithme Tabou a besoin de mémoriser la liste taboue pour éviter de cycler, la RT mémorise aussi tout au long de son évolution la meilleure solution S* rencontrée. En effet, à la fin de la recherche, lorsque le critère d'arrêt est vérifié, S* nous permettra de retrouver la meilleure solution visitée depuis le lancement de la recherche.

Par ailleurs, l'analyse de l'algorithme présenté dans cette section nous permet de déceler rapidement trois critères dont la configuration influe considérablement sur les résultats de la recherche en qualité et en temps. Le premier étant la taille de la liste taboue. Malgré l'apport positif que fournit cette liste, un mauvais paramétrage de sa longueur risque de lui faire perdre toute son utilité. D'un côté, plus la taille de cette liste est petite, plus le phénomène de cyclage reprend vie. D'un autre côté, une liste taboue trop longue limitera énormément l'espace de

recherche. Face à ce dilemme, il est généralement conseillé de procéder à une étude empirique dans l'optique de fixer la bonne taille de la liste taboue.

Le second élément à définir est la fonction V qui détermine le voisinage. La détermination de cette fonction a une grande influence sur l'évolution de la recherche. En effet, une mauvaise définition du voisinage pourrait ralentir considérablement l'évolution de l'algorithme. Donc la question à laquelle il faudra répondre pour définir le voisinage est : partant d'une solution S, comment choisir le sous ensemble des solutions S'∈ V(S)?

Finalement le troisième élément dont la configuration est assez fragile, est la condition d'arrêt. Ce dernier joue un rôle essentiel du point de vue de la qualité de la solution et du point de vue de son temps de réponse. On se retrouve ici face à un second dilemme : une condition d'arrêt rapidement atteignable ne laissera pas assez de temps au Tabou pour trouver l'optimum global recherché. À l'inverse, un critère d'arrêt peu probable prolonge trop longtemps la recherche même si l'optimum absolu est atteint.

Donc, afin de répondre aux besoins des annonceurs sur Internet, le module d'optimisation développé au cours de ce travail devra être capable de trouver une bonne solution dans un délai jugé raisonnable. Pour ce faire, nous proposons tout d'abord, d'implémenter l'algorithme de la recherche Tabou puisqu'il a démontré ses preuves dans plusieurs domaines d'application autant difficiles les uns que les autres [30]. Ensuite il faudrait passer par une phase de paramétrage de cet algorithme dont le but est d'améliorer ses performances.

2.3.2 Techniques d'améliorations

Dans cette section, nous présentons les trois principales techniques permettant d'améliorer les performances et l'efficacité de l'algorithme Tabou.

- **Critère d'aspiration**

Le critère d'aspiration fût introduit la première fois en 1986 par Fred Glover [26]. En 1989, il publie un autre article [28] sur son algorithme Tabou où il explique entre autre le fonctionnement du critère d'aspiration. L'idée derrière cette technique amélioratrice est d'accepter certains mouvements récemment effectués et qui en temps normal, ne seraient pas acceptés par les règles du Tabou. Le fait qu'on gère ici des mouvements plutôt que des solutions, nous permet de mémoriser uniquement ces mouvements au lieu de mémoriser des solutions complètes qui

alourdissent considérablement la recherche. Cette problématique est principalement rencontrée lors de la vérification de présence d'une solution voisine dans la liste taboue. Mais la mémorisation des mouvements ne présente pas que des avantages. Le fait de mémoriser des mouvements peut amener à visiter la même solution plusieurs fois et cela sans enfreindre l'interdiction qu'impose la liste taboue. Un autre inconvénient au fait de mémoriser des mouvements est qu'il se peut que la liste taboue ne permette pas de visiter des solutions qui n'ont jamais été explorées, même si celles-ci peuvent être des optimums. Face à ce dernier défaut relatif à la mémorisation des mouvements, l'utilisation du critère d'aspiration est d'une grande utilité. En effet, en libérant de la liste taboue uniquement les mouvements satisfaisant le critère d'aspiration, on peut trouver des solutions meilleures que la meilleure solution rencontrée depuis le début. Donc, le critère d'aspiration peut décider quand un mouvement est bénéfique et donne ainsi la permission à l'algorithme d'entreprendre ce mouvement même s'il est interdit.[29]

- **Technique de diversification**

Fred Glover nous explique bien cette technique dans son article [31]. En résumé ce processus consiste à effectuer des mouvements visant à varier l'échantillon de recherche. En effet il s'avère intéressant de diversifier les zones de recherches dans l'espoir de couvrir le plus possible de zones dans l'espace N. Généralement, cette technique est appelée lorsque la recherche se trouve bloquée aux alentours d'un optimum local. Grâce à la diversification, une solution sera générée, à partir de laquelle l'algorithme reprendra sa recherche.

- **Technique d'intensification**

Inversement à la technique de diversification, l'intensification effectue des mouvements permettant d'améliorer rapidement la solution. En effet, en favorisant la recherche aux alentours des zones les plus prometteuses, ce processus peut amener à des solutions de meilleure qualité. Comme son nom l'indique, cette technique intensifie et localise plutôt la recherche tout prêt des solutions avantageuses dans l'espoir de les améliorer et d'atteindre des solutions meilleures que S*.

CHAPITRE 3 : MODÉLISATION MATHÉMATIQUE : MÉTHODE EXACTE

Dans ce chapitre, nous commencerons par définir explicitement le problème à résoudre. Suite à cela, nous proposerons une modélisation mathématique de ce problème. Enfin, nous donnerons un aperçu des possibilités de résolution de ce problème par les méthodes exactes.

3.1 Définition du problème

Comme indiqué dans le paragraphe 1.3.3, plus une annonce est placée en haut de la liste des résultats d'un moteur de recherche, plus elle a des chances d'être cliquée. De ce fait, chaque annonceur veut voir son lien ou son annonce apparaître en haut de la page des résultats.

Dans le but de proposer une bonne gestion du portefeuille des annonceurs sur Internet, il est impératif de bien définir tout d'abord le problème auquel ils sont confrontés. Les clients, également appelés des annonceurs, veulent optimiser leurs campagnes publicitaires sur Internet. Pour cela, notre partenaire industriel leur offre une plateforme Web via laquelle ils pourront gérer chacune de leurs campagnes. L'objectif des annonceurs peut avoir plusieurs formes :

- Maximiser le nombre de conversions.
- Maximiser le nombre de clics.
- Maximiser le retour sur les dépenses en publicité « *Return on ad spend* ».
- Minimiser le coût total investi pour la publicité.

Les annonceurs cherchent également à atteindre ou à ne pas dépasser un seuil pour certains paramètres :

- Assurer un nombre de conversions minimal.
- Assurer un nombre de clics minimal.
- Assurer un nombre d'affichages minimal.
- Limiter les dépenses au budget prévu.
- Limiter le coût par annonce.
- Assurer une valeur minimale sur le retour sur les dépenses en publicité.

- Borner (min et / ou max) la position visée.
- Borner la valeur maximale du coût par clic.
- Taux moyen minimal de clics par impression.

À partir de ces deux listes, on sélectionnera un objectif et quelques contraintes, en vue de fixer un modèle sur lequel nous travaillerons tout au long de ce travail de maîtrise.

Pour chaque campagne publicitaire, les annonceurs devront fournir la liste des mots clés qui les intéressent. La taille de cette liste varie en général de 1 000 à 50 000 mots clés. Selon la requête associée à un mot clé donné, le site de l'annonceur peut apparaître à différentes positions sur la page des résultats de la recherche. Sachant que les positions les plus intéressantes sont celles de la première page (i.e. position 1 à 10) nous avons donc limité les positions possibles à 11. Tel que spécifié au chapitre 1, la position numéro 1 est la meilleure puisqu'on y observe en général un plus haut taux de clics, alors que ce taux décroît lorsque la position augmente. Compte tenu que les résultats s'étendent sur plus d'une page, les positions ne se limitent pas à 10. Toutefois, on observe pour les positions au-delà de 10 que le nombre d'impressions et de clics reste sensiblement le même. Compte tenu de ces observations et pour alléger notre modèle, nous considérons que la position 10 représente toutes les positions au-delà de 9. De plus, il est parfois intéressant d'exclure certains mots clés en ne prenant pas en compte leurs nombres de clics, d'impressions et leurs coûts. On propose donc une position fictive, représentée par la $11^{\text{ième}}$ position, afin de pouvoir mettre en pause un mot clé donné. Lorsqu'un mot clé est en pause, il n'est plus actif dans la campagne publicitaire.

Sachant que chaque mot clé peut prendre une position allant de 1 à 11, la taille de l'ensemble de recherche croit exponentiellement en fonction du nombre des mots clés à considérer. De ce fait, pour une campagne publicitaire qui compte n mots clés, la cardinalité de l'ensemble de recherche sera égale à 11^n.

En plus de la définition de cette liste, les annonceurs devront fixer le budget quotidien qu'ils sont prêts à dépenser par campagne. Suite à cela, ils devront également fixer la valeur d'enchère pour chacun des mots clés contenus dans leur liste. Pour cette dernière opération, il faut une grande expertise dans le domaine du Web marketing afin de mener à bien cette démarche.

3.2 Modélisation mathématique du problème

Le but de cette section est de présenter le modèle du problème à résoudre. Puisque plusieurs paramètres peuvent être considérés dans la recherche de la solution optimale, on commencera par donner la modélisation globale du problème.

Si aucune fonction non linéaire n'est utilisée, alors le problème peut prendre la forme générale du programme linéaire suivant :

$$Min\ (\ ou\ Max)\ Z = p_1x_1 + \ p_2x_2 + \cdots + \ p_nx_n$$

Sous contrainte :

$$\text{C1 :} \quad a_{11}x_1 + \ a_{12}x_2 + \cdots + \ a_{1n}x_n \left(\begin{matrix}\leqq \\ = \\ \geqq\end{matrix}\right) b_1$$

$$\text{C2 :} \quad a_{21}x_1 + \ a_{22}x_2 + \cdots + \ a_{2n}x_n \left(\begin{matrix}\leqq \\ = \\ \geqq\end{matrix}\right) b_2$$

…

$$\text{Cm :} \quad a_{m1}x_1 + \ a_{m2}x_2 + \cdots + \ a_{mn}x_n \left(\begin{matrix}\leqq \\ = \\ \geqq\end{matrix}\right) b_m$$

$$x_1, x_2, \dots, x_n \quad \in \ \{0\ ,1\}$$

Dans le but de comparer les résultats des différents tests, on est amené à fixer un exemple d'un modèle sur lequel on travaillera tout au long de ce travail de maîtrise. En général, les annonceurs Web souhaitent maximiser leurs profits. Puisqu'une conversion représente généralement l'action d'achat d'un client sur Internet, il serait alors intéressant de maximiser ce critère. Mais étant donné que dans certains cas les conversions sont difficilement dénombrables, nous proposons plutôt de retenir le nombre de clics comme indicateur de performance. Parmi les critères à respecter, le budget et un nombre minimum d'impressions sont ceux qui reviennent le plus souvent.

Donc, l'objectif qui sera utilisé dans notre modèle de base consiste à maximiser le nombre de clics. Cette optimisation se fera en tenant compte de deux types de contraintes : assurer un nombre minimal d'impressions et respecter la limite maximale sur le budget.

Compte tenu du choix de l'objectif et des contraintes, on peut présenter notre modèle de base sous la forme d'un programme linéaire en variables binaires.

❖ Définition des indices

i : Indique le numéro du mot clé.

j : Indique la position du mot clé

❖ Définition des paramètres

n : Constante définissant le nombre de mots clés.

$Budget$: Constante définissant le budget maximal à allouer.

Inf_Imp : Constante définissant le nombre minimal d'impressions désiré.

C_{Cost} , C_{Click} et C_{Imp} : Constantes relatives aux courbes génériques.

K_{Cost_i} , K_{Click_i} et K_{Imp_i} $\quad avec\ i \in \{1, \dots, n\}$: Constantes relatives aux courbes génériques.

$$Click_{ij} = \begin{cases} K_{Click_i} \times e^{(C_{Click} \times j)} & Pour\ j \in \{1, \dots, 10\} \\ 0 & Pour\ j = 11 \end{cases}$$

$$Cost_{ij} = \begin{cases} K_{Cost_i} \times e^{(C_{Cost} \times j)} & Pour\ j \in \{1, \dots, 10\} \\ 0 & Pour\ j = 11 \end{cases}$$

$$Imp_{ij} = \begin{cases} K_{Imp_i} \times e^{(C_{Imp} \times j)} & Pour\ j \in \{1, \dots, 10\} \\ 0 & Pour\ j = 11 \end{cases}$$

❖ Définition des variables

$$X_{ij} = \begin{cases} 1\ si\ le\ mot\ clé\ i\ est\ affecté\ à\ la\ position\ j \\ 0\ sinon \end{cases}$$

❖ Modèle de programmation linéaire

$$Max\ \mathcal{Z} = \sum_{i=1}^{n} \sum_{j=1}^{10} \left(Click_{ij} \times X_{ij}\right)$$

Sous contraintes:

$$C1 : \quad \sum_{i=1}^{n} \sum_{j=1}^{10} \left(Cost_{ij} \times X_{ij}\right) \leq Budget$$

$$C2 : \quad \sum_{i=1}^{n} \sum_{j=1}^{10} \left(Imp_{ij} \times X_{ij}\right) \geq Inf\_Imp$$

$$C3 : \quad \sum_{j=1}^{11} X_{ij} = 1 \qquad \forall\, i \in \{1, \ldots, n\}$$

$$C4 : \quad X_{ij} \in \{0, 1\} \qquad \forall\, i \in \{1, \ldots, n\}\, et\, \forall\, j \in \{1, \ldots, 11\}$$

La première contrainte C1 impose que la somme des coûts de tous les mots clés à une position donnée soit inférieure à un budget défini. C2 garantit que la somme des impressions générées à partir des positions affectées à chaque mot clé soit supérieure ou égale à un seuil défini. Quant à C3, elle exige que la somme de toutes les positions prises par un mot clé sera toujours égale à 1. Mais comme C4 précise que les variables X_{ij} sont tous binaires, alors on comprend des contraintes C3 et C4 qu'une seule position devra être sélectionnée pour tout mot clé. À première vue, ce type de problème semble facile à résoudre. Mais étant donné que les variables sont binaires, le problème devient de plus en plus complexe à résoudre au fur et à mesure que le nombre de mots clés augmente.

En plus du modèle linéaire présenté précédemment, on introduit un second modèle qui est de type quadratique. La seule différence entre les deux modélisations s'inscrit dans la formulation de la fonction objectif. En effet, dans le but de tester notre module d'optimisation dans le cas non linéaire, on a eu recourt à une seconde formulation de la fonction objectif qui se présente comme suit :

$$\mathcal{M}ax\ F = \sum_{i=1}^{n} \sum_{j=1}^{10} \left(Click_{ij} \times\ Imp_{ij}\ \times X_{ij}^{2}\right)$$

Étant donné que les variables X_{ij} sont binaires, X_{ij}^{2} sera toujours égale à X_{ij}. Ceci nous amène à dire que cette formulation de la fonction objectif est artificielle. Mais puisque notre objectif est de tester les performances de notre module d'optimisation dans le cas non linéaire, on se permet d'accepter une telle formulation.

3.3 Résolution du modèle par une méthode exacte

3.3.1 Mise en place des paramètres nécessaires à la résolution

Pour résoudre le modèle mathématique décrit à la section 3.2 avec une méthode exacte, on aura besoin tout d'abord d'un solveur assez puissant qui pourra trouver la solution optimale. La figure 3.1 décrit tout le système qui doit être mis en place afin que le solveur puisse fonctionner correctement. En effet, pour que le solveur puisse trouver la solution optimale, il faudra lui fournir nécessairement le modèle mathématique qu'on souhaite résoudre. En plus, il aura besoin des données de base qu'on présente sous la forme des courbes génériques à la figure 3.1. Enfin, avant de lancer la résolution, il faudra fixer les différentes constantes définissant les contraintes du problème et indiquer au solveur le nombre de mots clés *n*. En contrepartie, le solveur devra trouver la solution optimale et la présenter sous la forme d'un vecteur des positions optimales de chacun des mots clés.

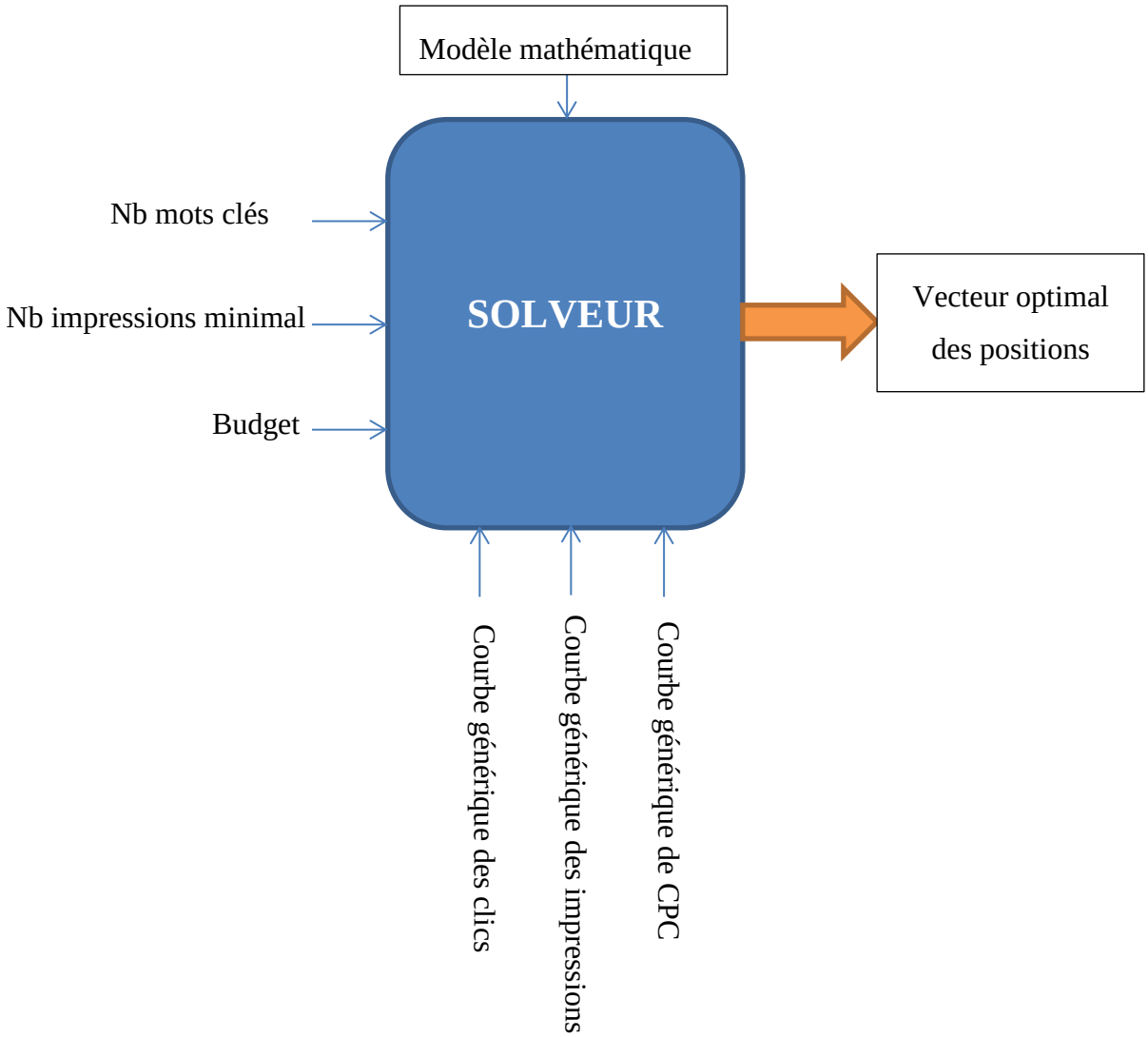

Figure 3-1 : Schéma du système de résolution

Pour pouvoir trouver la solution optimale, le solveur aura besoin des données historiques de la base de données de notre partenaire industriel. Comme il est expliqué dans le chapitre de la revue de littérature, nous nous basons dans ce travail sur les courbes génériques développées par Patrick Quinn [21] afin de prédire les nombres de clics, d'impressions, ainsi que le coût généré suite à l'affectation d'un mot clé à une position donnée.

Grâce à ces fonctions génériques exponentielles, nous sommes capables d'estimer le nombre de clics en appliquant la formule : $clics = k_{clics} * e^{m*pos}$, où m est un paramètre qui caractérise la forme de la décroissance de la courbe et k_{clics} représente la valeur (nombre de clics) à la position 1. Ces courbes donneront une estimation du nombre de clics pour une position donnée. Il faut noter que la courbe générique des clics est utilisée ici à titre d'exemple et qu'on a les mêmes courbes de prédiction pour le cas de CPC et des impressions.

Afin de faciliter l'exploitation des courbes génériques pour le solveur et même pour le programme de l'algorithme Tabou, on propose de calculer en prétraitement, pour chacun des mots clés et pour toutes ses positions possibles (de 1 à 11), le nombre de clics, le nombre d'impressions et le coût. La matrice (*n* x P) qui sera générée pour chaque caractéristique suite à ces calculs servira à attribuer les coefficients aux variables dans la fonction objectif ainsi que dans les contraintes.

Il est important de mentionner que dans le cadre de ce mémoire de maîtrise, les courbes impressions versus positions ou encore CPC versus position proviennent de travaux antérieurs (i.e. le mémoire de maîtrise de P. Quinn entre autres) [21]. Il n'est pas dans le mandat de ce mémoire de juger de la qualité de l'information fournie par ces courbes. Nous les utiliserons dans un contexte où il sera aisé pour les recherches futures de les changer si de meilleures estimations devenaient disponibles.

3.3.2 Résolution par la méthode exacte

Tout au long de ce travail de maîtrise, nous utiliserons le solveur CPLEX 12.5 qui permet de trouver la solution optimale avec une méthode de résolution exacte. Initialement cet outil permet de résoudre des problèmes de programmation linéaire contenant plusieurs dizaines de milliers de variables et plusieurs centaines de milliers de contraintes [33]. Afin de trouver la solution optimale, CPLEX est doté d'une implémentation performante du simplexe primal. Mais dépendamment du type de problème, CPLEX choisit l'algorithme approprié qui sera suivi éventuellement d'un algorithme de séparation et d'évaluation progressive dans le cas où le problème contient des variables entières [33]. Au cours de ce travail nous utiliserons la solution optimale donnée par CPLEX afin de la comparer à la solution qui sera produite par la recherche Tabou.

Pour que CPLEX puisse fonctionner, il aura besoin de deux fichiers. Le premier étant une description du modèle mathématique selon la syntaxe OPL de CPLEX. Et le deuxième n'est autre qu'un fichier de données contenant l'information associée aux fonctions génériques des clics, d'impressions et des coûts par clic sous la forme matricielle. En plus, ce fichier de données devra fournir :

- Le nombre total des mots clés *n*,
- le budget à ne pas dépasser,
- le nombre minimal d'impressions.

Une fois que toute cette phase de paramétrage et de programmation dans CPLEX est menée à bien, nous sommes capables de résoudre le problème par la méthode exacte. La solution optimale qui est produite par CPLEX est constituée de 4 éléments :

- Le nombre total des clics,
- le nombre total d'impressions,
- le coût total,
- un vecteur de dimension *n* qui contient la position de chaque mot clé.

Tout au long de ce chapitre, on a décrit tout le processus dont on s'est servi afin de résoudre le problème par une méthode exacte. Les résultats qui nous seront fournis à partir de ce processus serviront à décider quant à la qualité des résultats qui seront dévoilés par l'algorithme Tabou qui sera présenté lors du chapitre suivant.

CHAPITRE 4 : IMPLÉMENTATION DE L'ALGORITHME TABOU

Le quatrième chapitre de ce mémoire est consacré à l'implémentation de l'algorithme Tabou pour l'optimisation des campagnes publicitaires sur Internet. Le module d'optimisation à développer pour les différents tests du chapitre 5 devra maximiser le nombre de clics tout en respectant une contrainte budgétaire et tout en garantissant un nombre d'impressions. Alors, le programme devra faire appel, d'une part aux différentes techniques de l'algorithme de la recherche Tabou, et d'autre part, à la base de données historique de notre partenaire industriel. Il y a aussi plusieurs éléments dans l'algorithme Tabou qui doivent être convenablement déterminés et précisés afin que cet algorithme soit correctement implémenté. Pour ce faire, on expose dans chacune des sections de ce chapitre les divers éléments composant l'algorithme.

Par ailleurs, il est intéressant de mentionner qu'afin d'implémenter l'algorithme, nous codons avec le langage de programmation orienté objet C++ sous l'environnement de développement *Code Blocks 12.11*.

4.1 Définition des solutions

Dans cette section, nous présentons la forme générale des solutions. Ensuite, nous expliquerons le rôle de chacune des différentes solutions spécifiques qui ont été utilisées lors de l'implémentation du programme.

4.1.1 Forme générale des solutions

Grâce à l'aspect orienté objet du C++, chaque solution est composée de cinq éléments :

- Score : un réel à maximiser. Il est calculé pour chaque solution et est égal au nombre de clics dans le cas où toutes les contraintes sont respectées. Le calcul du score est détaillé dans la section 4.4.
- Nombre de clics : un nombre réel contenant la somme des clics générés à partir des différentes positions des mots clés. L'estimation du nombre de clics moyen de chaque mot clé provient de la courbe générique des impressions en fonction des positions.

- Nombre d'impressions : un nombre réel contenant la somme des impressions relatives aux différentes positions des mots clés.
- Coût total : c'est un nombre réel qui estime le coût total moyen en calculant la somme des coûts générés suite à la position prise par chacun des mots clés. Le coût généré de chaque mot clé est calculé quant à lui, en faisant la multiplication entre le CPC généré et le nombre de clics généré d'un même mot clé.
- Vecteur position : Un tableau unidimensionnel de *n* cases contenant chacune un entier (position) entre 1 et 11. On fait correspondre la première case du tableau au premier mot clé et ainsi de suite jusqu'au nième mot clé auquel est consacrée la nième place du vecteur position.

4.1.2 La solution initiale

Soit S_0 la solution initiale de la recherche Tabou. C'est à partir de cette solution de départ que le Tabou commencera à explorer divers voisinages jusqu'à atteindre d'autres solutions de meilleure qualité. Dans le programme développé au cours de cette maîtrise, on se propose de tester trois différentes solutions de départ. La liste suivante décrit chacune de ces solutions :

- La pire solution S_0': On génère dans la solution de départ un vecteur de positions en mettant chacun des mots clés à la position 11. Cette solution de départ n'est pas réalisable vu que son nombre d'impression est nul et ainsi elle ne garantit pas le nombre d'impression minimal.
- Solution améliorée S_0'' : Cette fois-ci, afin de générer la solution initiale, un prétraitement s'impose. Le pseudocode décrivant ce prétraitement est le suivant :

1. Trier la liste des mots clés selon un ordre décroissant du nombre de clics
2. Pour i allant de 1 à n
3. pour j allant de 1 à 11
4. si la contrainte du budget n'est pas violée, alors
5. le mot clé i prend la position j

- Solution relaxée S_0''' : Dans ce dernier cas d'étude, une nouvelle approche de génération de S_0 est mise à l'épreuve. Puisque les logiciels d'optimisations tels que CPLEX ou GLPK (logiciel libre) fournissent de bons résultats, alors il serait intéressant de lancer l'algorithme Tabou à partir de la solution issue de ces logiciels en relaxant les contraintes d'intégralité. Puisque la solution (vecteur positions) issue de la relaxation linéaire contient des positions fractionnelles, alors il faudra arrondir ces positions à l'entier supérieur afin de ne pas violer la contrainte du budget.

4.1.3 La solution S_k

Vu que la recherche Tabou est une recherche itérative, on note S_k la solution courante à l'itération k. Une solution S_k est dite réalisable lorsque son coût total est inférieur à un budget fixé et que son nombre d'impressions est supérieur à un nombre d'impressions minimal fixé lui aussi.

4.1.4 La solution S*

La solution S* est la meilleure solution réalisable rencontrée depuis le début de la recherche. À chaque itération, le programme vérifie si S_k est une solution réalisable et si le score $F(S_k)$ de S_k est plus grand que celui de S*. Si ces deux conditions sont satisfaites, S* est mis à jour avec le contenu de la solution S_k.

4.1.5 La solution S_{Best}

Cette solution est très semblable à S*. La seule différence entre ces deux solutions est que S_{Best} peut contenir une solution non réalisable. Donc, l'algorithme compare uniquement le score de la solution courante à celui de S_{Best}. Si le score de l'une des solutions du voisinage est plus grand que le score de S_{Best}, alors cette dernière sera mise à jour. L'idée derrière S_{Best} est de poursuivre encore la recherche aux alentours de cette bonne solution malgré sa non réalisabilité dans l'espoir de trouver une solution réalisable S* tout près de S_{Best}.

4.2 Déplacement

D'une itération à une autre, l'algorithme Tabou apporte des changements minimes à la solution courante S_k. On appelle ces changements élémentaires, des déplacements ou des mouvements. Ainsi, partant de la solution initiale S_0, l'algorithme va se déplacer progressivement d'une solution à une autre, jusqu'à atteindre généralement des solutions de meilleure qualité. À la figure 4.1, on schématise un exemple de déplacement avec des solutions ayant 5 mots clés. On découvre à partir de la figure 4.1 que durant un déplacement, il n'y a que le quatrième mot clé qui change de position dans le vecteur des positions. Dans cet exemple, c'est le mot clé numéro 4 qui passe de la position 6 à la position 2. Ce déplacement engendre naturellement un changement du score, du nombre de clics moyen, du nombre d'impressions moyen et bien évidemment du coût.

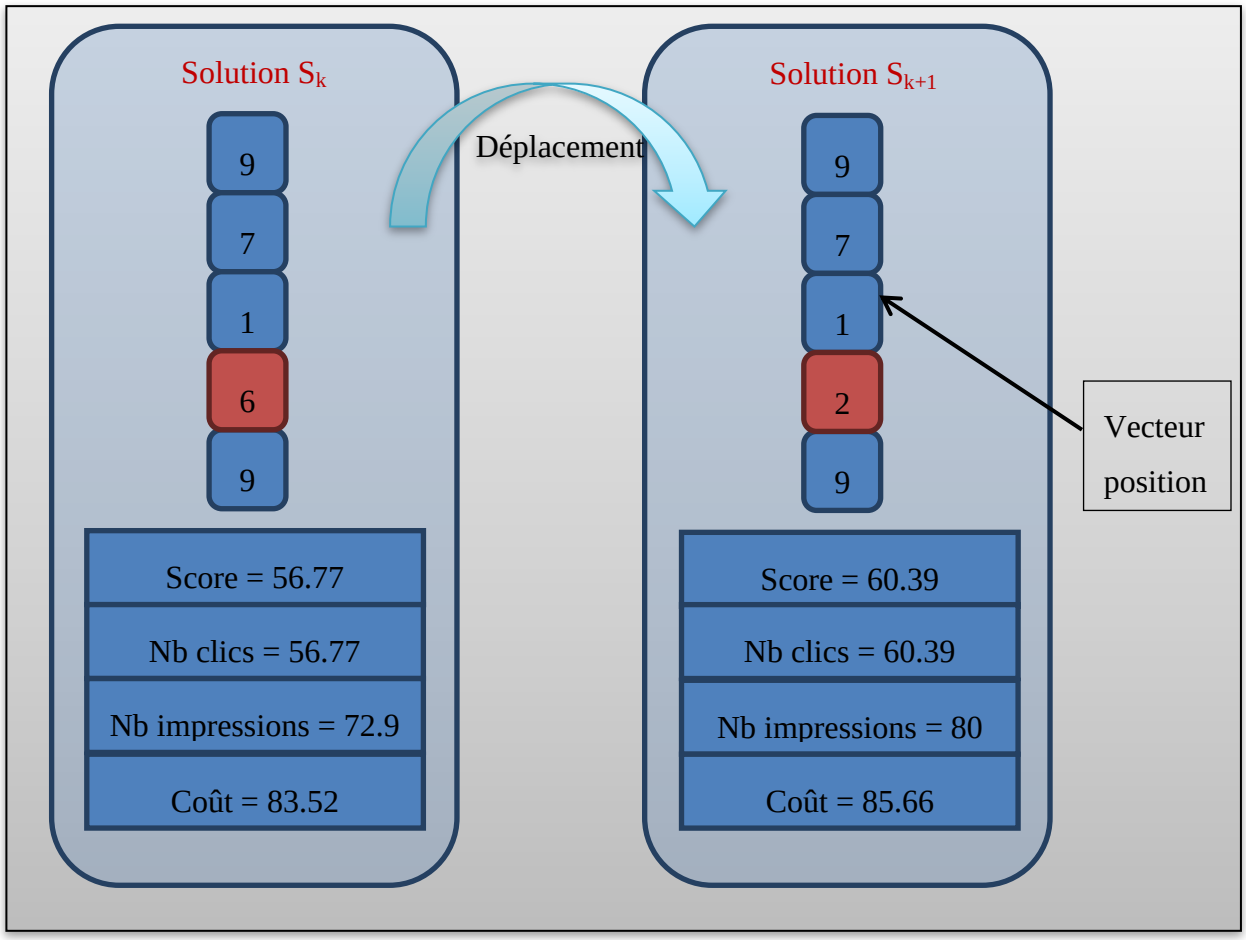

Figure 4-1: Exemple d'un déplacement avec des vecteurs de 5 mots clés

4.3 Voisinage

4.3.1 Génération du voisinage

La taille maximale du voisinage qui est généré à chaque itération est proportionnelle au nombre *n* des mots clés. En effet, pour construire le voisinage, on parcourt aléatoirement tous les mots clés de la solution courante et on essaye de mettre chacun d'eux à chacune des dix autres positions possibles. Ainsi, pour la résolution d'un problème contenant *n* mots clés, on a un voisinage de taille maximale égale à 10 *n*.

4.3.2 Sélection du voisin

Pour des raisons de mémoire et de temps de calcul, le voisinage ne sera pas totalement généré durant chaque itération. Le programme génère un voisin S_v puis l'évalue immédiatement avant de générer le prochain voisin. Si le score de ce voisin est meilleur que celui de S*, et si de plus ce voisin S_v respecte les contraintes C1 et C2, alors on copie S_v dans S* et on interrompt la génération du voisinage.

Sinon on vérifie si la solution S_v n'est pas taboue et trois situations se présentent :

- Si la position actuelle du mot clé à changer est supérieure à 6 et que la nouvelle position de ce même mot clé est inférieure à 6, alors l'algorithme peut passer à la phase d'évaluation du score de cette solution.
- Si la position actuelle du mot clé à changer est inférieure à 6 et que la nouvelle position de ce même mot clé est supérieure à 6, alors l'algorithme peut passer à la phase d'évaluation du score de cette solution.
- Sinon dans le cas où la position actuelle du mot clé à changer est égale à 6, alors toutes les autres positions du même mot clé peuvent être acceptées.

Le processus de génération du voisinage sera stoppé au premier voisin non tabou qui améliore le score de S_{Best}. Dans le cas échéant, l'algorithme n'aura d'autre choix que de parcourir tous les voisins et d'en sélectionner le meilleur voisin non tabou, même s'il détériore le score de la solution courante.

4.4 Fonction objectif

La modélisation de la fonction objectif dans le programme de l'algorithme Tabou diffère en forme de celle du modèle LP décrite dans la section 3.2. Afin de permettre à l'algorithme d'explorer des chemins assez critiques, passant par des solutions non réalisables, on assouplit les contraintes C1 et C2 du modèle LP. Dans cette optique, on insère dans la fonction objectif deux variables d'écarts e_{Cost} et e_{Imp} qui vont pénaliser toute violation de C1 et C2, respectivement. Afin de pouvoir réguler l'amplitude des variables d'écarts, on associe à e_{Cost} le paramètre de pondération β et on associe à e_{Imp} le paramètre de pondération γ. La formule suivante représente la fonction objectif utilisée dans le programme.

$$\mathcal{M}ax\ F = \sum_{i=1}^{n} \sum_{j=1}^{11} (Click_{ij} \times X_{ij}) - \beta\, e_{Cost} - \gamma\, e_{Imp}$$

Avec :

$$e_{Cost} = max\left\{ 0\ ,\quad \sum_{i=1}^{n} \sum_{j=1}^{11} (Cost_{ij} \times X_{ij}) - Budget \right\}$$

$$e_{Imp} = max\left\{ 0\ ,\quad Inf\_Imp - \sum_{i=1}^{n} \sum_{j=1}^{11} (Imp_{ij} \times X_{ij}) \right\}$$

$\beta \geq 0\ et\ \gamma \geq 0$

Avec cette modélisation du problème, l'algorithme Tabou pourra emprunter des chemins passant par des solutions non réalisables, mais tout en gardant en mémoire dans S*, la meilleure solution réalisable rencontrée.

4.5 Paramètres de pondérations

Dans le but de contrôler et de diriger l'évolution de la recherche, tantôt violant les contraintes et tantôt les respectant, on introduit dans la fonction objectif deux paramètres de régulation. Dans la fonction objectif présentée à la section 4.4, on remarque la présence de deux

paramètres de pondérations β et γ. Ces deux paramètres ont été insérés dans la fonction d'évaluation afin que nous puissions réguler l'intensité des deux variables d'écarts e_{Cost} et e_{Imp}. En effet, lors d'une violation du budget, le score se verra diminué de $(\beta \times e_{Cost})$. Alors, plus β est grand, plus l'impact de la variable d'écart e_{Cost} dans la fonction objectif est accentué et vice-versa. Le même raisonnement s'applique pour le paramètre γ qui amplifie l'impact de e_{Imp} dans la fonction objectif. Ainsi, en fonction des valeurs attribuées à β et γ, l'algorithme pourra assouplir ou restreindre le respect des contraintes C1 et C2.

Afin que l'algorithme puisse orienter la recherche efficacement, il faut développer un mécanisme qui fera varier les valeurs des paramètres de pondération β et γ de façon automatique. Cette méthode a été expliquée par Gendreau et Al. dans leur article paru en 1994 [32]. Pour ce faire, on implémente une méthode qui varie les valeurs de β et γ selon l'évolution de la recherche. Pour le cas de β, la fonction de régulation devra compter le nombre de violations successives ou de respects successifs de la contrainte C1. Une fois que ce nombre de violations ou de respects successifs de C1 atteint une limite définie, la valeur de β sera doublée ou divisée par deux afin d'augmenter l'impact, ou d'assouplir celui-ci, lorsqu'il y aura violation de la contrainte C1. Une procédure similaire est développée pour le paramètre γ afin de contrôler le niveau de relaxation de la contrainte C2.

Ainsi on est parvenu à implémenter la fonction d'autorégulation des paramètres de pondération. Toutefois, afin d'éviter que la fonction d'autorégulation des paramètres β et γ ne fasse croître la valeur de ces paramètres indéfiniment, on propose de créer des seuils qui limitent la croissance de β et γ. Ces limites sont elles aussi variables. Cependant, le mécanisme gérant ces limites parvient à restreindre la probable évolution infinie de β et γ. L'algorithme suivant explique le processus de la mise à jour du seuil de β. Un algorithme similaire existe pour la mise à jour du seuil de γ.

Définition des paramètres :

- β_{inf} : La borne inférieure de β
- β_{sup} : La borne supérieure de β

- compteur_beta_div : Compteur du nombre de fois successives où β doit baisser en dessous de sa borne inférieure β_{inf}.
- compteur_beta_mul : Compteur du nombre de fois successives où β doit dépasser sa borne supérieure β_{sup}.

1. Si ($\beta/2 < \beta_{inf}$) et que la valeur de β doit être divisée par deux
2. Alors incrémenter compteur_beta_div
3. Sinon réinitialiser compteur_beta_div
4. Si ($\beta \times 2 > \beta_{sup}$) et que la valeur de β doit être multipliée par deux
5. Alors incrémenter compteur_beta_mul
6. Sinon réinitialiser compteur_beta_mul
7. Si (compteur_beta_div = 50)
8. Alors β_{inf} := $\beta_{inf}/2$ et réinitialiser compteur_beta_div
9. Si (compteur_beta_mul = 50)
10. Alors β_{sup} := $\beta_{sup} \times 2$ et réinitialiser compteur_beta_mul

4.6 Liste taboue T

4.6.1 Forme de la liste taboue

Plusieurs modélisations de la liste taboue sont offertes. Dans le module d'optimisation que nous implémentons, la liste taboue se présente sous la forme d'une matrice comme le montre la figure 4.2. Les éléments t_{ij} ($i \in \{1, \dots, n\}$ et $j \in \{1, \dots, 11\}$) de cette matrice T indiquent à l'algorithme le numéro de l'itération à partir de laquelle le mouvement sera permis.

	Pos1	Pos2	Pos3	Pos4	Pos5	Pos6	Pos7	Pos8	Pos9	Pos10	Pos11
MC1											
MC2											
MC3				39							
MC4											
MC5											

Figure 4-2 : Forme générale de la liste taboue avec $n = 5$

Dans la figure 4.2, chaque ligne de la matrice taboue correspond à un mot clé. Par contre, les colonnes de cette matrice représentent les différentes positions que peuvent prendre les mots clés. La colonne de la sixième position (colonne en rouge) est considérée comme le seuil séparant les basses (i.e. moins bonnes) positions de 7 à 11 et les hautes (i.e. les meilleures) positions de 1 à 5. Cette différenciation entre les basses et les hautes positions n'a de l'importance que lors de la mise à jour de la liste taboue. Ce point sera expliqué plus en détail à la section 4.6.2

Afin de faciliter la compréhension du fonctionnement de cette modélisation de la liste taboue, on présente un exemple à la figure 4.2. On parvient à déterminer le laps de temps durant lequel le mouvement demeurera tabou en faisant la somme du numéro de l'itération courante et de la durée courante du statut tabou. Dans la figure 4.2, on peut lire dans la case (MC3 , Pos4) le nombre 39. Ce nombre indique que le mot clé numéro 3 était à la position 4 et l'a quitté à l'itération 33. En supposant dans cet exemple que la durée du statut tabou étant de 6, alors en faisant la somme du numéro de l'itération courante 33, avec la durée courante du statut tabou 6, on retrouve le nombre 39. Ce nombre indique que le mouvement défini par la mise du mot clé 3 à la position 4 est tabou tant que le numéro de l'itération courante est inférieur ou égal à 39. Si le compteur indiquant l'itération courante est supérieur à la valeur de l'élément t_{ij} de la matrice taboue, alors le déplacement de la position du mot clé i à la position j est permis. Sinon ce dernier mouvement est considéré comme tabou.

4.6.2 Taille de la liste taboue

La taille de la liste taboue est considérée comme l'un des facteurs de succès d'une bonne implémentation de l'algorithme Tabou. Afin de déterminer ce paramètre, toute une série de tests a été mise en place. On décrit ces différents tests dans le cinquième chapitre. Initialement, on a fixé la taille de cette liste égale à la racine carrée de la taille du voisinage comme le propose Bouchard et al. [34]. Mais comme la taille du voisinage réellement généré n'est pas connue d'avance et varie d'une itération à une autre, alors on a dû créer un compteur qui détermine le nombre de voisins qui ont été générés durant chaque itération. Ensuite, en arrondissant à l'entier supérieur la racine carrée du nombre des voisins et en ajoutant à ce nombre un chiffre aléatoirement tiré entre -5 et +5, on trouve la taille de la liste taboue pour l'itération courante. La formule suivante explique comment la longueur de la liste taboue est calculée à chaque itération :

$$Taille\_Tabou = \left\lceil \sqrt{Taille\_Voisinage} \right\rceil + Aléa_{[-5\ ,\ 5]}$$

Ce processus est répété durant chaque itération afin de calculer la taille de la liste taboue.

4.6.3 Mise à jour de la liste taboue

La mise à jour de la liste taboue suit les trois cas de figure qui ont été présentés dans la section 4.3.2 et voici comment cette liste se met à jour :

- Si l'ancienne position du mot clé à changer est supérieure à 6, alors on interdit durant un nombre d'itérations correspondant à la taille (longueur) de la liste taboue actuelle, de repositionner le mot clé courant aux positions 7 à 11.
- Si l'ancienne position du mot clé à changer est inférieure à 6, alors on interdit durant un nombre d'itérations correspondant à la taille de la liste taboue actuelle, de repositionner le mot clé courant aux positions 1 à 5.
- Sinon dans le cas où l'ancienne position du mot clé à déplacer est égale à 6, alors on interdit durant un nombre d'itérations correspondant à la taille (longueur) de la liste taboue actuelle le retour à cette position.

Cette méthodologie permet de limiter les déplacements inutiles en essayant de privilégier les déplacements qui sont les plus rentables.

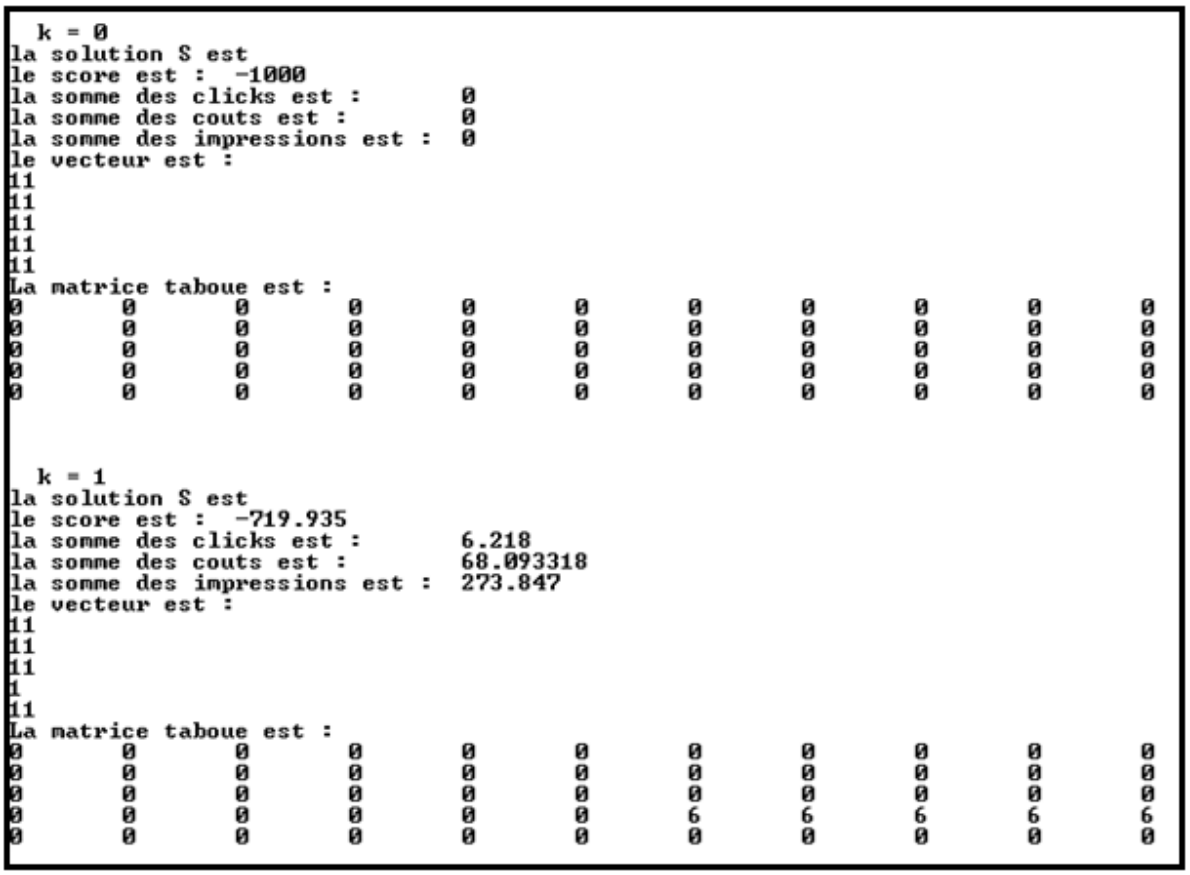

Figure 4-3 : Représentation du premier cas de figure de la mise à jour de la liste taboue

On présente aux figures 4.3, 4.4 et 4.5 des exemples de l'exécution du programme principal avec n = 5. Ces trois exemples schématisent respectivement les trois cas de figures modélisant le processus de mise à jour de la liste taboue.

À la figure 4.3, on explique le premier cas de mise à jour de la liste taboue. Le déplacement présenté dans cette figure est le tout premier de l'algorithme Tabou. Initialement quand le compteur k est à zéro, le vecteur de S_0 est initialisé à 11 et la liste taboue contient des zéros partout. Donc aucun déplacement n'est interdit durant cette première itération. Quand k passe à un, on remarque dans le vecteur de S1 que le quatrième mot clé est passé de la position 11 à la position 1. Donc comme on peut le voir dans la matrice taboue pour k = 1, les positions 7 à 11 du quatrième mot clé sont tous mis à 6 où 6 est la taille de la liste taboue calculée à la première itération.

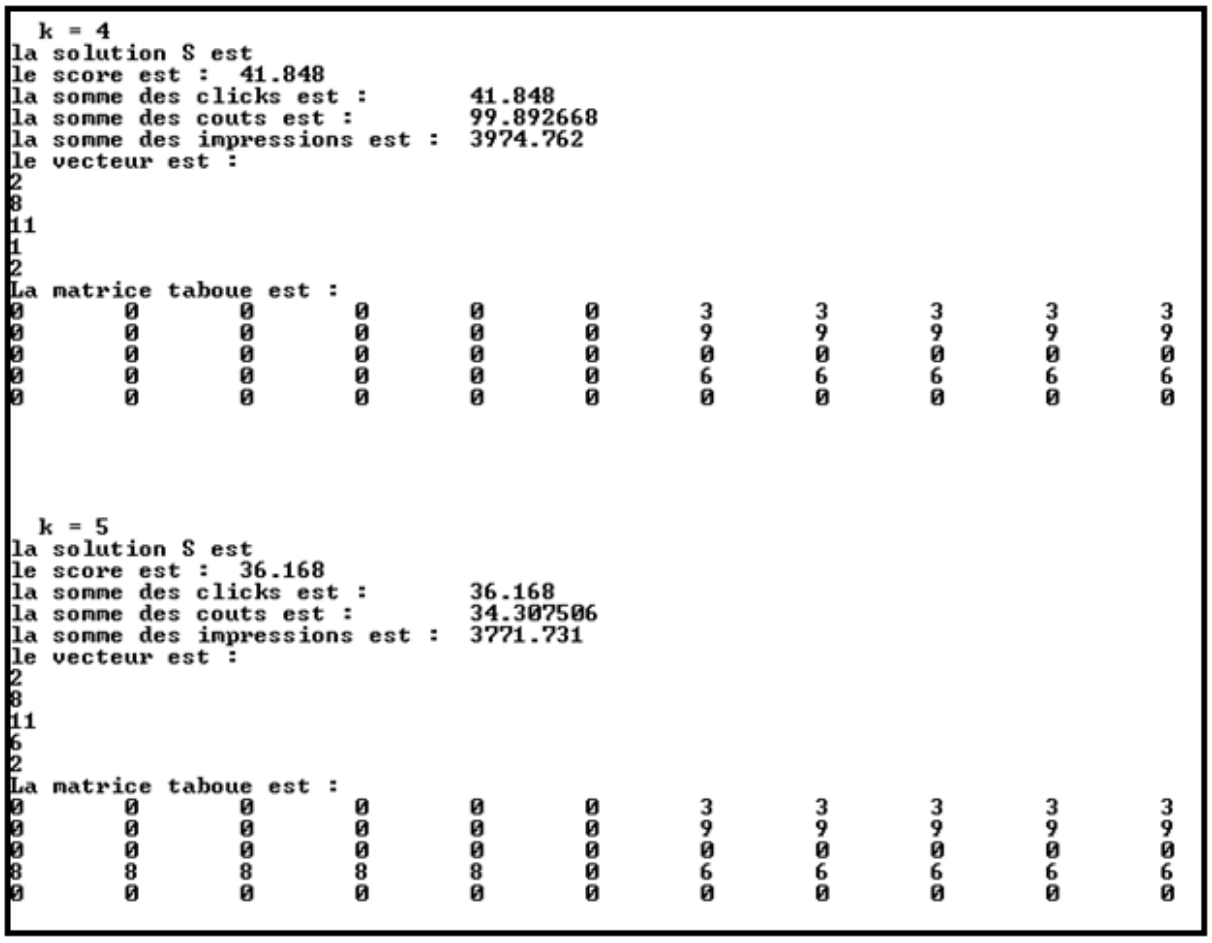

Figure 4-4 : Représentation du deuxième cas de figure de la mise à jour de la liste taboue

À la figure 4.4 on expose le deuxième cas de la mise à jour de la liste taboue. De l'itération 4 à l'itération 5, on remarque que le quatrième mot clé passe de la position 1 à la position 6. Puisque l'ancienne position (i.e. à l'itération k=4) de ce mot clé est inférieure à 6, alors on se retrouve dans le deuxième cas de figure énoncé au début de cette section. De ce fait, on voit que les positions 1 à 5 de la quatrième ligne de la matrice taboue sont mises à 8. Ceci se traduit par l'interdiction des cinq premières positions au quatrième mot clé tant que le compteur k n'a pas atteint la huitième itération.

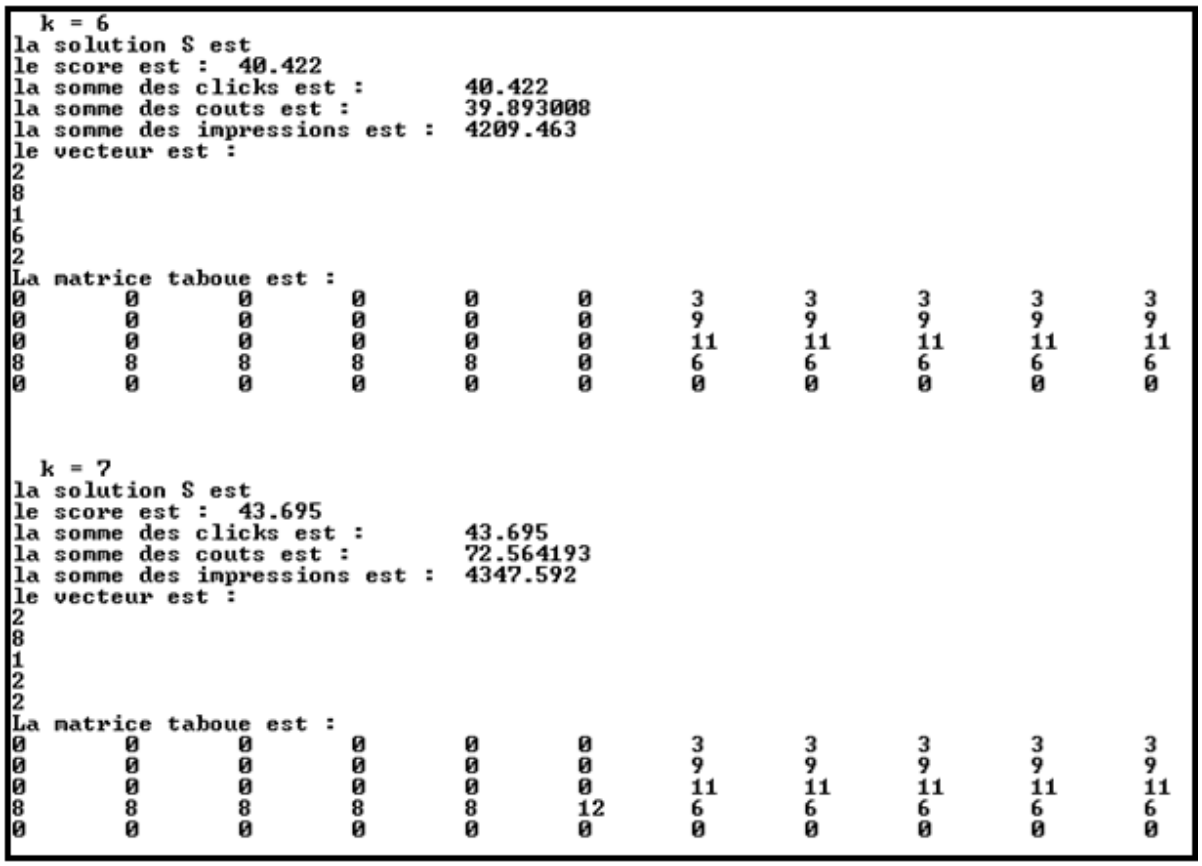

Figure 4-5 : Représentation du troisième cas de figure de la mise à jour de la liste taboue

Dans ce dernier cas de figure, on remarque que le quatrième mot clé passe de la position 6 à l'itération 6, à la deuxième position à l'itération 7. Par conséquent, une seule case de la matrice taboue devrait être modifiée selon les règles de la mise à jour de la liste taboue qu'on a présentées. En effet, à la figure 4.5, on remarque que le déplacement de l'itération 6 à 7 a engendré la mise à jour de la quatrième ligne et la sixième colonne de la matrice taboue pour passer de 0 à 12. Donc la position 6 sera interdite au mot clé numéro 4 tant que l'algorithme n'a pas atteint l'itération 12.

4.7 Critère d'aspiration

Comme dans toute bonne implémentation de l'algorithme Tabou, il est indispensable d'avoir un critère d'aspiration qui a le pouvoir d'enfreindre les interdictions de la liste taboue. Dans notre cas, le critère d'aspiration se manifeste à chaque fois que l'un des voisins respectant

les contraintes C1 et C2 améliore le score de S*. Même si ce voisin est présent dans la liste taboue, notre critère d'aspiration lui permet d'être choisi.

4.8 Technique de diversification

Rappelons que l'objectif de la recherche est de maximiser le nombre de clics tout en respectant une contrainte liée au budget qui ne doit pas être dépassé et en garantissant un nombre défini d'impressions. À travers les différents tests du programme on s'est aperçu que le nombre d'impressions minimal est rapidement atteint et ne cause donc pas de vrai problème. Par contre, en voulant maximiser le nombre de clics, il est évident que l'algorithme essaierait de positionner le plus possible les mots clés aux premières positions. Comme les premières positions sont généralement les plus coûteuses, plus le nombre de clics va augmenter, plus le coût total croitra. Donc l'algorithme se trouvera face à un dilemme : d'un côté il veut maximiser le nombre de clics et d'un autre côté le dépassement du budget pénaliserait de plus en plus le score.

Partant de cette constatation, on a cherché à savoir ce qui se passerait si on désactivait la contrainte du budget durant un certain nombre d'itérations puis on la réactivait ultérieurement. Grâce au paramètre de pondération β, il est devenu facile de désactiver la contrainte du budget. En effet, en forçant le paramètre β à zéro, l'algorithme de recherche n'est plus retenu par la contrainte du budget et du coup, le score de S croît constamment tant que β est nul. La figure 4.6 schématise ce phénomène à partir de l'itération 2200 (trait en rouge) jusqu'à l'itération 2600 qui est limitée par un trait vert sur la figure.

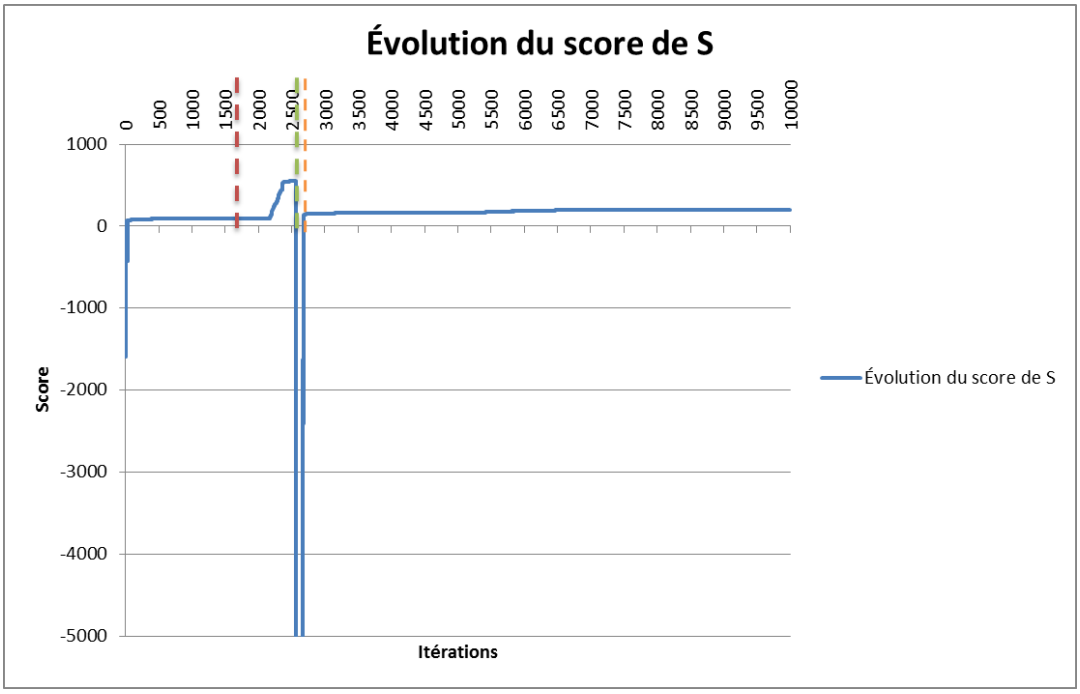

Figure 4-6 : Courbe de l'évolution du score de S pour les premières 10000 itérations

Durant ce délai, on assiste aussi à une montée excessive du coût total de la solution courante comme le montre la courbe de la figure 4.7. Après un certain nombre d'itération où β est à zéro (400 itérations dans l'exemple), on découvre que la zone de recherche est totalement différente, ce qui est le but principal de la technique de diversification.

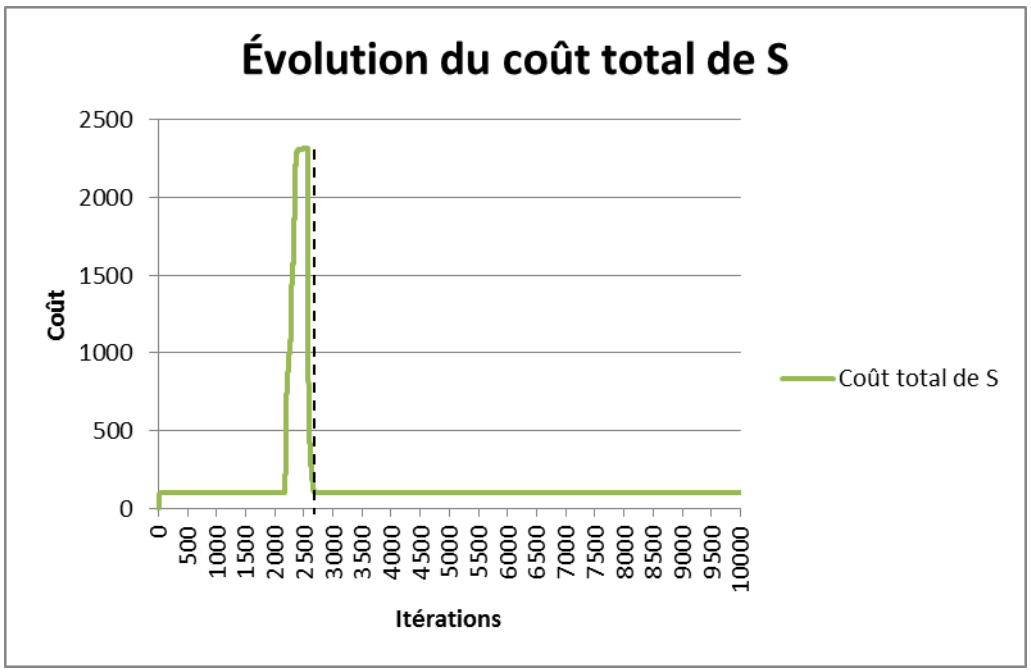

Figure 4-7 : Courbe de l'évolution du coût total de la solution courante pour les premières 10 000 itérations

Maintenant qu'une stratégie de diversification a été proposée, il faut déterminer le moment propice pour la lancer. En effet, afin de savoir à quel moment il serait approprié de lancer soit la technique de diversification ou encore la technique d'intensification qui sera exposée à la section 4.9, on fait appel à un second compteur qu'on note Z. Ce nouveau compteur est réinitialisé à chaque fois que l'une des solutions S* ou S_{Best} est mise à jour. Ainsi, ce nouveau compteur Z nous indique le nombre d'itérations passées sans amélioration de S* ou S_{Best}. Donc, en exploitant ce compteur, on serait capable de savoir si la recherche évolue dans le bon sens ou bien si elle peine à améliorer le score.

En résumé, la technique de diversification est appelée au bout d'un certain nombre d'itérations successives sans amélioration du score. Une fois que le compteur Z atteint ce nombre défini préalablement, on met à zéro le paramètre de pondération β durant un certain nombre d'itérations jugé suffisant pour que la recherche puisse changer de voisinage. Enfin, une fois que le délai alloué à la diversification est écoulé (dans cet exemple 400 itérations), on remet le paramètre β à une grande valeur pour qu'il puisse pénaliser fortement le score. La figure 4.6 nous permet de voir cette pénalisation entre les traits en vert et en orange. En effet, à partir de l'itération 2600, le paramètre β est réactivé avec la valeur de 100 ce qui pénalise

considérablement le score qui chute en dessous de zéro. Ainsi, l'algorithme sera automatiquement forcé à reprendre la recherche dans des zones où le budget n'est pas grossièrement violé. Cette reprise qui concorde avec la fin de la diversification peut être remarquée à partir de l'itération 2600 de la courbe de la figure 4.6. Par ailleurs, l'effet de cette reprise se distingue clairement dans la courbe de l'évolution du score de S* (figure 4.8). À partir de cette figure, on constate que l'algorithme Tabou n'arrive plus à améliorer le score de S* de l'itération 1200 jusqu'à la fin de diversification. Effectivement, de K = 1200 à K = 2200 (début de la diversification) le compteur Z aurait atteint les 1000 itérations successives sans amélioration du score. On débute alors la procédure de diversification pour débloquer la recherche. Après 400 itérations durant lesquelles la technique de diversification tente de changer de zone de recherche, on assiste à des améliorations successives de score à partir de l'itération 2700 environ.

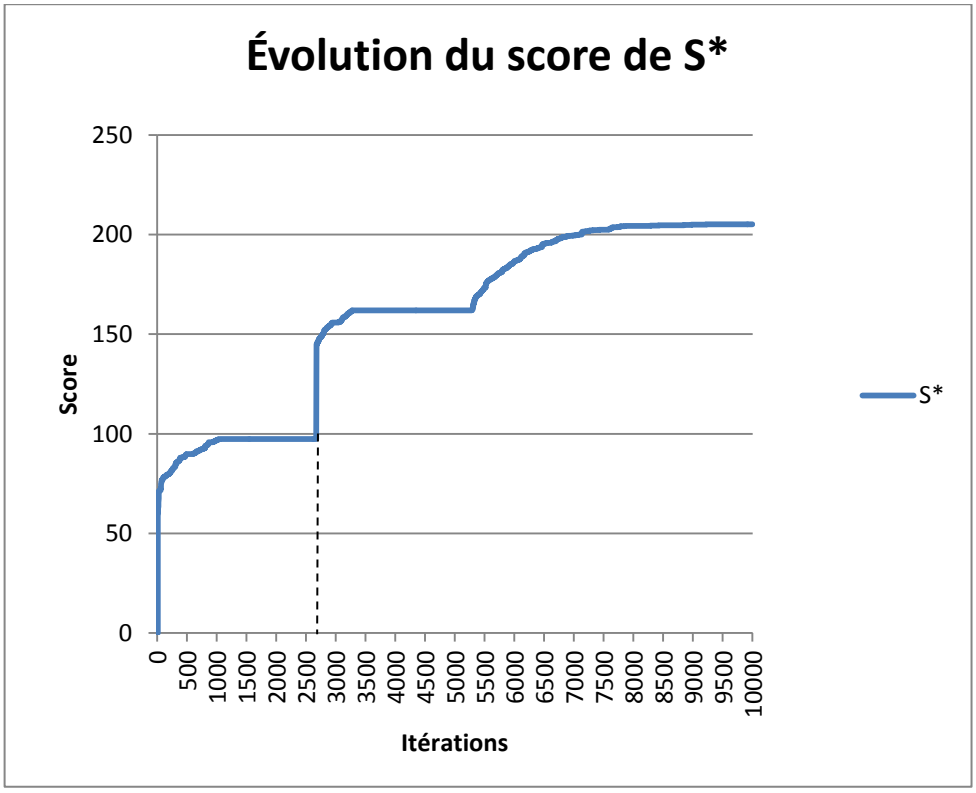

Figure 4-8 : Courbe de l'évolution du score de S* pour les premières 10000 itérations

4.9 Technique d'intensification

Les techniques d'intensification sont généralement appelées à la fin du processus de recherche puisque le but de ces techniques est d'essayer d'améliorer la meilleure solution obtenue. Encore une fois, le compteur Z sera utilisé afin d'estimer à quel moment il faudra faire appel à une technique d'intensification.

Durant ce travail de maîtrise, deux différentes techniques d'intensifications ont été développées. Chacune d'elles a fait ses preuves durant la phase de test du programme développé. De ce fait, elles sont toutes les deux utilisées dans le programme final. À la figure 4.9, on distingue une nette reprise des améliorations du score à partir de l'itération 5300 où les techniques d'intensification sont actionnées. Leurs effets sont clairement remarquables à la figure 4.9 puisque la courbe d'évolution du score de S* s'est remise à croître à partir de cette itération.

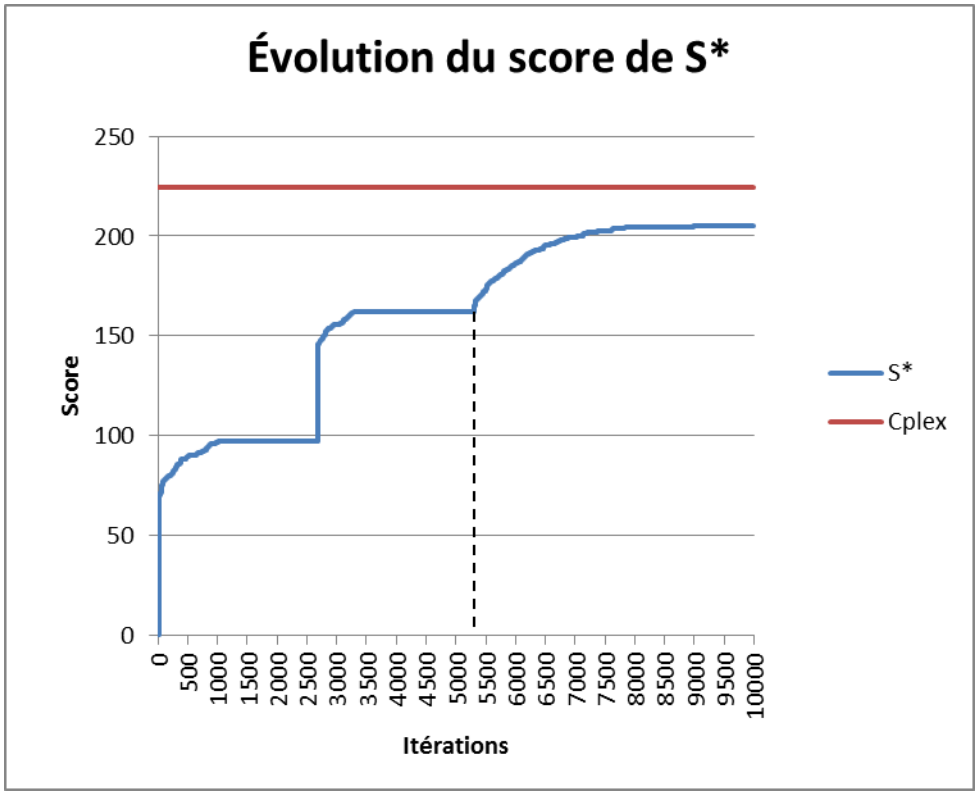

Figure 4-9 : Comparaison de la courbe de l'évolution du score de S* pour les 10000 premières itérations avec le score optimal donné par CPLEX

On remarque aussi, à partir de cette figure, qu'après l'activation des techniques d'intensifications, la recherche Tabou commence à s'approcher de plus en plus de la solution optimale donnée par CPLEX (courbe en rouge).

4.9.1 Première technique d'intensification

La première technique d'intensification consiste à sélectionner aléatoirement un mot clé pour le mettre à la position 11. Puis en revenant au fonctionnement habituel de la recherche Tabou, l'algorithme décidera s'il est avantageux de remettre ce mot clé à sa place ou bien s'il est plus payant de déplacer un autre mot clé. Cette technique ne doit être bien évidemment appelée qu'à la fin, lorsque la recherche ne parvient plus à améliorer le score de S*. Toute l'idée derrière ce mécanisme est qu'en mettant en pause un mot clé donné, son coût sera déduit du coût total de toute la solution. Ainsi, on crée une marge de manœuvre à l'algorithme pour qu'il puisse remonter les positions de certains mots clés sans un inévitable dépassement du budget. Cette technique peut être appelée autant de fois que possible tant que le score s'améliore.

4.9.2 Deuxième technique d'intensification

Lors de la comparaison entre le vecteur position donné par CPLEX et celui trouvé par l'algorithme Tabou, on a remarqué que la majorité des mots clés occupent exactement la même position. À quelques exceptions près, la plupart des discordances entre ces deux vecteurs sont minimes. En effet, les mots clés qui occupent des positions différentes ont généralement un écart d'une seule position. Cette deuxième technique d'intensification a été créée afin de tenter de corriger ces discordances.

Cette seconde technique d'intensification parcourt aléatoirement la liste des mots clés et en sélectionne deux à chaque fois. Ensuite, comme l'indique la figure 4.10, elle recule le premier mot clé (coloré en rouge) d'une position et avance l'autre (coloré en vert) d'une position. Enfin, elle réévalue le score de cette nouvelle solution obtenue et la compare au score de S* après s'être assurée que la nouvelle solution S*' respecte les contraintes. Si le score de S*' est meilleur que celui de S*, alors l'objectif est atteint.

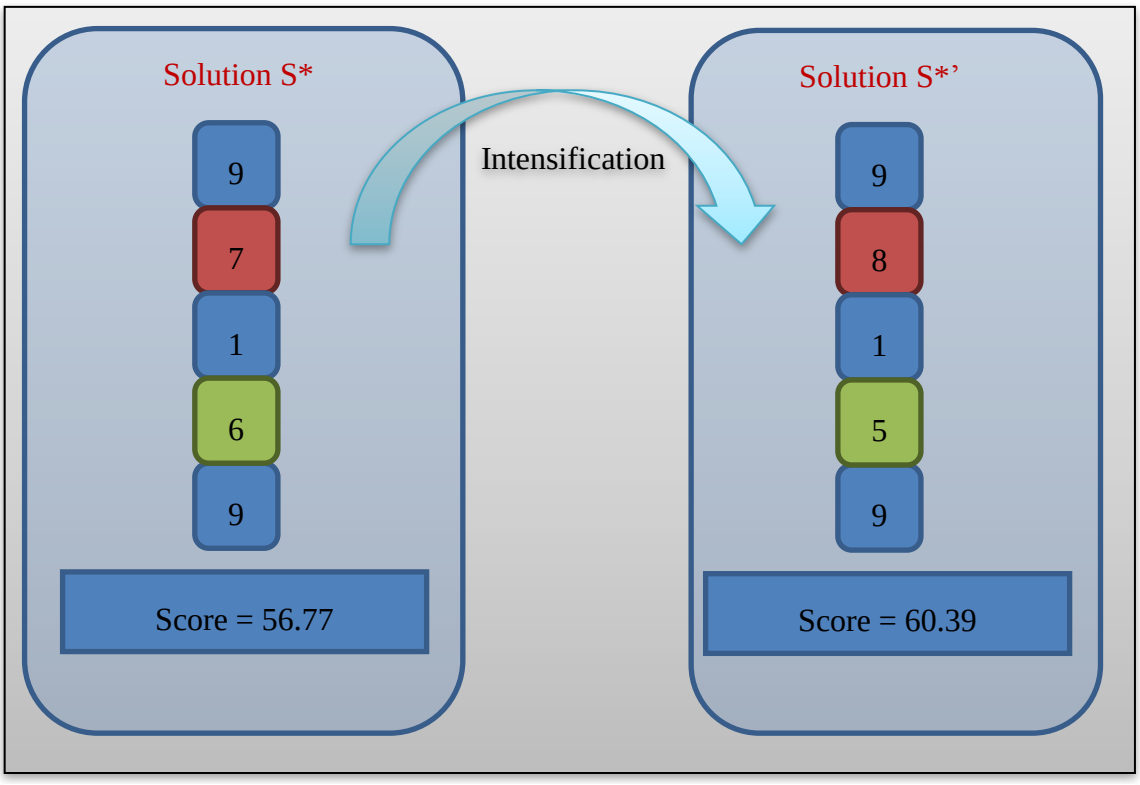

Figure 4-10 : Exemple de la deuxième technique d'intensification

Dans ce quatrième chapitre, on a expliqué l'implémentation de l'algorithme de la recherche Tabou. Lors du chapitre suivant, nous allons présenter les différents résultats de tests qui nous ont permis de fixer certains paramètres de l'algorithme et enfin nous dévoilerons les résultats finaux, fournis par le programme implémenté.

CHAPITRE 5 : AMÉLIORATIONS DE LA RECHERCHE TABOU ET ANALYSE DES RÉSULTATS

Après avoir exposé dans le quatrième chapitre tous le processus de développement de l'algorithme Tabou, on présente dans ce cinquième et dernier chapitre les différents résultats des tests ainsi que les résultats finaux du module d'optimisation. Dans la première section, on présente les différents tests réalisés et leurs résultats afin de pouvoir fixer certains paramètres de l'algorithme Tabou tels que la taille de la liste taboue et autres. Dans la seconde et dernière section de ce chapitre, on présente les résultats finaux auxquels on a abouti grâce au module d'optimisation développé au cours de cette maîtrise.

5.1 Tests pour le paramétrage de l'algorithme Tabou

5.1.1 Présentation des séries de tests

À travers des tests préliminaires de l'évolution de l'algorithme Tabou, on estime que si la recherche atteint le seuil de 5000 itérations successives sans avoir amélioré le score de S*, alors il y a de fortes chances que la recherche en cours s'est piégée dans un maximum local dont elle ne peut s'évader. C'est à cet instant que la technique de diversification devra être invoquée, afin de libérer la recherche courante du maximum local en l'orientant vers un sous-espace encore inexploré.

Partant de cette constatation, on se fixe comme seuil le nombre de 5000 itérations successives sans amélioration de la meilleure solution. Alors pour pouvoir suivre l'évolution de l'algorithme Tabou sans faire appel à aucune technique d'amélioration, on arrêtera l'exécution du programme après 5000 itérations successives sans amélioration du score de S*.

Deux grands scénarios, qu'on nommera S1 et S2, sont mis à l'épreuve. Les deux différences principales entre le scénario S1 et le scénario S2 touchent essentiellement à la façon dont les déplacements d'une solution vers la suivante se produisent, et la façon dont la liste taboue est mise à jour suite au choix de la solution S_{k+1} vers laquelle on se déplacera.

- Scénario 1 : à chaque itération, on interdit à l'aide de la liste taboue l'état défini par le mot clé M_{k+1} et la position Pos_k. M_{k+1} étant le mot clé dont la position va être modifiée dans la solution S_{k+1}. La position Pos_k est la position prise par le mot clé M_{k+1} dans la solution S_k.

- ❖ Scénario 2 : à chaque itération, on interdit les états définis par le mot clé M_{k+1} dont sa position va être modifiée dans la solution S_{k+1}, et la ou les positions qui sont définis comme suit :
  - De la position 1 à 5, si ($Pos_k < 6$ et $Pos_{k+1} > 6$)
  - De la position 7 à 11, si ($Pos_k > 6$ et $Pos_{k+1} < 6$)
  - La position 6 si ($Pos_{k+1} = 6$)

Pour chacun de ces deux scénarios, on va essayer de changer à chaque nouveau test l'un des paramètres P décrits ci-dessous, puis on interprètera et analysera son impact sur l'évolution de l'algorithme Tabou :

- P0 : Configuration de base.
- P1 : Augmentation de la taille de la liste taboue (multiplication par deux).
- P2 : Diminution de la valeur de mise à jour de la pénalité β (de $\beta/2$ à $\beta/5000$).
- P3 : Augmentation du seuil maximal de la pénalité β (de 1.29 à 500).
- P4 : Augmentation du nombre de violations successives ou bien de respects successifs de la contrainte relative au coût, avant que beta ne soit mise à jour (de 10 à 500).

Pour pouvoir interpréter les résultats, on propose qu'à chaque fin de test, on récupère le numéro de l'itération atteint et bien-sûr le score de S*. Grâce à ces deux indicateurs de performances, il serait possible de sélectionner la meilleure configuration de l'implémentation de l'algorithme Tabou.

Les résultats des tests relatifs aux paramètres P1, P2, P3 et P4 seront comparés à la configuration de base (P0) et ceci pour chacun des deux scénarios S1 et S2. Donc, pour un scénario donné, si l'une des configurations testées atteint un score supérieur à celui de P0 alors la variation du paramètre relatif à cette configuration est prometteuse. Dans certains cas où le score de deux configurations distinctes ne varie pas, on compare les nombres d'itérations de chacune de ces configurations. Cela nous permet de savoir si la variation du paramètre en question améliore ou pas la rapidité de l'algorithme.

5.1.2 Résultats des tests

Dans ce paragraphe, on présente les résultats des tests du programme développé. Afin que l'interprétation de ces résultats soit la plus cohérente possible, on a choisi de varier l'échantillon à tester en taille et en contenu. En effet, il est présenté dans ce qui suit, le résultat de six bases de données distinctes : deux ayant la taille de 500 mots clés, deux autres de 1000 mots clés et les deux dernières bases contiennent 5000 mots clés.

On présente les résultats de chacune des bases de données dans un tableau à part. Afin de bien distinguer la différence entre les résultats du scénario 1 et du scénario 2, on a choisi de réserver une colonne pour chacun des deux scénarios. Dans la troisième colonne on présente le score optimal donné par CPLEX comme le montre le tableau 5.1. Ce score optimal est présenté dans chacun des tableaux 5.1, 5.2, 5.3, 5.4, 5.5 et 5.6, uniquement pour donner une idée sur la qualité des scores atteints par chaque configuration. En ce qui concerne les lignes, elles sont formées des différentes configurations à tester. Pour chaque configuration P, on expose trois critères : le score de la meilleure solution S*, le pourcentage de rapprochement du score de S* par rapport au score de CPLEX et enfin le nombre d'itérations pour atteindre le score de S*.

- **Base de données 1**

Tableau 5.1 : Résultat de BD 1

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	204,096	197,381	224,451
	Taux de S*	91%	88%	
	Nb itérations	2055	10907	
P1	Score de S*	203,929	196,672	
	Taux de S*	91%	88%	
	Nb itérations	4568	11824	
P2	Score de S*	192,496	193,218	
	Taux de S*	86%	86%	
	Nb itérations	4911	1517	
P3	Score de S*	208,234	195,41	
	Taux de S*	93%	87%	
	Nb itérations	4867	7358	
P4	Score de S*	206,675	187,661	
	Taux de S*	92%	84%	
	Nb itérations	5760	5016	

À partir des résultats récupérés dans le tableau 5.1, on remarque que le scénario 1 l'emporte du point de vue du score sur le scénario 2. S2 a obtenu un meilleur score que celui de S1 (dans le cas de P2) une seule fois mais cette différence est assez minime. Par ailleurs, on remarque que les paramètres P3 et P4 ont un impact positif puisqu'ils améliorent nettement le score, comparé à celui de P0 qui est la configuration de base.

Donc jusqu'à présent, ce premier test favorise le scénario 1 en augmentant le seuil maximal de la pénalité β et en tolérant plus de violations successives, avant de mettre à jour la variable de pondération β.

- **Base de données 2**

Tableau 5.2 : Résultat de BD 2

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	$-\infty$	$-\infty$	3940,589
	Taux de S*	-	-	
	Nb itérations	540	1687	
P1	Score de S*	$-\infty$	$-\infty$	
	Taux de S*	-	-	
	Nb itérations	539	4250	
P2	Score de S*	$-\infty$	$-\infty$	
	Taux de S*	-	-	
	Nb itérations	540	1687	
P3	Score de S*	2211,666	$-\infty$	
	Taux de S*	56%	-	
	Nb itérations	27478	81	
P4	Score de S*	$-\infty$	$-\infty$	
	Taux de S*	-	-	
	Nb itérations	499	499	

Les choses sont un peu plus complexes pour cette base de données. Effectivement, les résultats du tableau 5.2 démontrent que l'algorithme Tabou éprouve même de la difficulté à trouver une solution réalisable. Le seul cas où la RT est parvenue à trouver une solution réalisable c'était dans le cas du scénario 1 et lorsqu'on augmente le seuil maximal de la pénalité β.

❖ **Base de données 3**

Tableau 5.3 : Résultat de BD 3

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	3815,188	3745,188	29859,485
	Taux de S*	13%	13%	
	Nb itérations	1026	2321	
P1	Score de S*	3815,188	3745,188	
	Taux de S*	13%	13%	
	Nb itérations	1026	3422	
P2	Score de S*	3815,188	3745,188	
	Taux de S*	13%	13%	
	Nb itérations	1006	2280	
P3	Score de S*	18414,626	17351,632	
	Taux de S*	62%	58%	
	Nb itérations	14180	14515	
P4	Score de S*	3815,188	3745,188	
	Taux de S*	13%	13%	
	Nb itérations	1003	1003	

Dans le cas de cette base de données, l'analyse des résultats du tableau 5.3 indique qu'il n'y a pas vraiment une différence significative entre les scores donnés par le scénario S1 et le scénario S2. La plupart du temps les scores de S1 et de S2 sont à 13% du score optimal.

On peut également constater à partir du tableau 5.3, que P3 est encore le seul paramètre qui a engendré une amélioration du score. En augmentant le seuil maximal de β, le score passe de 13% d'écart par rapport au score de la solution optimale à 62% pour le cas de S1 et de 13% à 58% dans le cas du scénario S2.

❖ **Base de données 4**

Tableau 5.4 : Résultat de BD 4

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	1095,773	1095,773	19731,416
	Taux de S*	6%	6%	
	Nb itérations	5649	2851	
P1	Score de S*	1095,773	1095,773	
	Taux de S*	6%	6%	
	Nb itérations	3579	14296	
P2	Score de S*	1095,773	1095,773	
	Taux de S*	6%	6%	
	Nb itérations	5649	2851	
P3	Score de S*	10574,908	11531,926	
	Taux de S*	54%	58%	
	Nb itérations	16978	18639	
P4	Score de S*	1095,773	1095,773	
	Taux de S*	6%	6%	
	Nb itérations	999	999	

Pour cette quatrième base de données, on retrouve les mêmes constatations que celles tirées des tests de la troisième base de données. De plus, on remarque d'après le tableau 5.4 que pour le paramètre P4, même si le score n'a pas changé, on arrive tout de même à l'atteindre beaucoup plus rapidement (un nombre d'itérations plus petit). Cette dernière remarque s'applique aussi pour le cas du scénario S1 et pour le paramètre P1. Ces deux dernières remarques suggèrent que l'augmentation de la taille de la liste taboue pourrait représenter un paramètre favorisant l'amélioration du programme du point de vue de la rapidité.

❖ **Base de données 5**

Tableau 5.5 : Résultat de BD 5

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	1487,614	1487,614	27221,607
	Taux de S*	5%	5%	
	Nb itérations	24154	18319	
P1	Score de S*	1487,614	1487,614	
	Taux de S*	5%	5%	
	Nb itérations	35230	12622	
P2	Score de S*	1487,614	1487,614	
	Taux de S*	5%	5%	
	Nb itérations	23068	17984	
P3	Score de S*	25990,834	16438,643	
	Taux de S*	95%	60%	
	Nb itérations	28779	46528	
P4	Score de S*	1487,614	1487,614	
	Taux de S*	5%	5%	
	Nb itérations	2512	2512	

Dans ces tests réalisés sur la cinquième base de données, on constate d'après le tableau 5.5 que seul le paramètre P3 influe sur le score de S*. Il est vrai qu'au cours de ce test le nombre d'itérations a augmenté. Mais cela s'explique par le fait que l'algorithme Tabou aura besoin de beaucoup plus d'itérations pour parvenir à un score nettement meilleur que celui donné dans le cas de P0.

On constate aussi que du point de vue de la rapidité de convergence, le scénario S2 l'emporte sur S1 au cours des tests effectués sur P0, P1 et P2.

❖ **Base de données 6**

Tableau 5.6 : Résultat de BD 6

		Scénario 1	Scénario 2	CPLEX
P0	Score de S*	1119,35	1544,943	1901,842
	Taux de S*	59%	81%	
	Nb itérations	763	12522	
P1	Score de S*	1146,37	1249,454	
	Taux de S*	60%	66%	
	Nb itérations	851	4833	
P2	Score de S*	919,211	1039,654	
	Taux de S*	48%	55%	
	Nb itérations	490	514	
P3	Score de S*	710,183	811,551	
	Taux de S*	37%	43%	
	Nb itérations	174	222	
P4	Score de S*	1758,747	46,437	
	Taux de S*	92%	2%	
	Nb itérations	37293	1533	

Dans le cas de cette base de données, on remarque que le scénario 2 l'emporte sur le scénario 1 dans la plupart des cas. En effet dans P0, P1, P2 et P3 le scénario S2 donne de bien meilleurs scores. Certes, dans les cas cités précédemment, S2 prend plus de nombre d'itérations pour générer la solution. Mais à ce stade du projet, on privilégie le facteur de la qualité de la solution sur le paramètre du temps de réponse. En plus, il est tout à fait logique que dans le cas de S2 le Tabou affiche un nombre d'itération plus grand que celui de S1 vu que la RT est une recherche itérative et que plus il passe de temps à chercher, plus il a de chances d'améliorer le score de S*. Le plus important à ce stade c'est que la recherche ne soit pas bloquée rapidement dans une solution de mauvaise qualité.

D'après le tableau 5.6, on constate aussi que les paramètres P0 dans le cas de S2 et P4 dans le cas de S1 présentent d'assez bons résultats. Ceci prouve que l'augmentation du seuil maximal de la pénalité β n'est pas toujours une bonne action, puisque dans P0 on a un score meilleur que celui dans P3.

Par contre, une augmentation du nombre de violations successives ou bien de respects successifs de la contrainte relative au coût, avant que β ne soit mise à jour est une bonne action à entreprendre. Effectivement, le tableau 5.6 indique que dans le cas de S1, le test P4 donne un score proche à 92 % du score optimal calculé par CPLEX. Cette dernière constatation va à l'encontre des résultats de toutes les séries de tests des bases de données précédentes.

5.1.3 Synthèse de l'interprétation des résultats

Une fois que les résultats des différents tests ont été présentés, il est temps maintenant de synthétiser les interprétations de la section 5.1.2 afin de conclure quant à la meilleure configuration de l'algorithme Tabou.

Pour ce faire, nous présentons dans le tableau 5.7 la synthèse de tous les tableaux exposés dans le paragraphe 5.1.2. Le tableau 5.7 résume tous ces résultats en donnant une vue globale de ces derniers. Au lieu d'afficher les scores et les nombres d'itérations numériquement, on représente par une flèche croissante ou décroissante l'évolution des résultats du test.

Tableau 5.7 : Synthèse des résultats

		P 1		P 2		P 3		P 4	
		Score	Nb itérations	Score	Nb itérations	Score	Nb itérations	Score	Nb itérations
BD 2440	S1	↘	↗	↘	↗	↗	↗	↗	↗
	S2	↘	↗	↘	↘	↘	↗	↘	↘
BD 2509	S1	–	–	–	–	↗	↗	–	–
	S2	–	–	–	–	–	–	–	–
BD 3589	S1	–	–	–	↘	↗	↗	–	↘
	S2	–	↗	–	↘	↗	↗	–	↘
BD 3609	S1	–	↘	–	–	↗	↗	–	↘
	S2	–	↗	–	–	↗	↗	–	↘
BD 4169	S1	–	↘	–	↘	↗	↗	–	↘
	S2	–	↘	–	↘	↗	↗	–	↘
BD 4229	S1	↗	↗	↘	↘	↘	↘	↗	↗
	S2	↘	↘	↘	↘	↘	↘	↘	↘

Clé	
↘	Dégradation par rapport à P0
↗	Amélioration par rapport à P0
–	Pas de changement

On résume dans ce tableau les résultats des 60 tests qui ont été réalisés sur 6 différentes bases de données. Partant d'une configuration de base P0, on compare les résultats de chacun des tests P1, P2, P3 et P4 aux résultats donnés par P0. On remarque que dans certain cas il y une amélioration ou augmentation du score et que dans d'autres, il y a une dégradation ou diminution du score toujours par rapport à ceux obtenus par la configuration de base. On récupère les résultats des tests précédents et on essaie de les présenter d'une manière plus simple et globale afin de pouvoir mieux les interpréter.

Il est important de signaler que le nombre d'itérations au bout duquel l'algorithme Tabou s'est arrêté est aussi un indicateur qui ne manque pas d'importance. En effet, même si le score n'a pas été changé, il est fort intéressant d'atteindre le même score mais en un nombre d'itérations

moindre. Cela se révèle un point assez important quant à la vitesse de convergence de l'algorithme.

À partir de ce tableau 5.7, on sera capable de répondre aux 5 questions pour lesquelles tous ces tests ont été mis en place. Rappelons-les dans ce qui suit :

1. Quel serait le scénario le plus prometteur : S1 versus S2?
2. Est-ce que le paramètre P1, qui teste l'effet de l'augmentation la taille de la liste taboue sur les résultats, serait une bonne action ou pas?
3. Est-ce qu'une diminution de la valeur de mise à jour de la pénalité β améliorerait le score final?
4. Une élévation du seuil maximal de la pénalité β est-elle nécessaire pour améliorer le rendement du Tabou?
5. Enfin, l'augmentation du nombre de violations successives ou bien de respects successifs de la contrainte relative au coût, avant que β ne soit mise à jour aurait-il un bon impact sur l'efficacité de l'algorithme Tabou?

❖ **Choix du scénario : S1 versus S2**

Dans la plupart des tests réalisés, on note que le scénario S1 donne de meilleurs résultats que ceux atteints par S2. Toutefois, S2 arrive dans certain cas à surpasser S1. Par exemple, d'après le tableau 5.7, on remarque que dans le cas de la troisième base de données et pour le paramètre P3, S2 atteint le même score que S1. Par contre S2 y parvient en un nombre d'itération moindre. Ce critère ne manque pas d'importance dans la phase de l'amélioration des performances de l'algorithme Tabou, puisqu'un nombre d'itérations plus petit implique un temps d'exécution plus cours.

S2 améliore non seulement le temps de réponse mais aussi la qualité du score. En effet, les résultats des tests rapportés dans la première ligne du tableau 5.6 appuient cette constatation, en dévoilant le score de 1119,35 pour S1 et un score de 1544,943 pour S2.

Puisque les deux scénarios procurent de bons résultats, il est important de conserver les deux. Alors la seule façon de bénéficier de la contribution des deux scénarios en même temps est de fusionner ces deux mécanismes dans le même algorithme.

Pour ce faire, dans la version finale de l'algorithme Tabou, on choisira parmi le voisinage, la première solution rencontrée qui est réalisable et ayant un score meilleur que celui de la solution S*. On ne vérifiera pas si elle est taboue ou pas, puisqu'une amélioration du score de S* implique que la solution n'a jamais été visitée. Dans ce cas de figure la recherche dans le voisinage sera arrêtée.

Dans le cas où le parcourt de tout le voisinage se termine sans avoir trouvé aucune solution qui réponde aux critères précédents, le programme choisira la meilleure solution de tout le voisinage même si elle dégrade le score de S*. Par contre dans ce cas, le mouvement à effectuer ne devra pas figurer dans la liste taboue et la mise à jour de cette liste se fera selon le scénario S2.

Avec cette nouvelle configuration de l'implémentation de l'algorithme Tabou, on arrive à bien fusionner les deux scénarios afin d'en tirer le maximum de profit.

- **Sélection du paramètre P1**

Afin de bien fixer la taille de la liste taboue, on propose comme point de départ, la racine carrée de la taille du voisinage. Ce rapport entre la taille de la liste taboue et la taille du voisinage fut l'objet d'un bon nombre de publications scientifiques. Selon Bouchard et al. (2009) [34], pour les tests préliminaires, une bonne longueur de la liste taboue serait de la fixer à la racine carrée du voisinage.

On remarque à partir des tests effectués au cours de ce chapitre, qu'une augmentation de la taille de la liste taboue à deux fois la racine carrée influe considérablement sur les performances de l'algorithme. En effet, la première colonne P1 du tableau 5.7 montre que dans certains cas cette augmentation améliore le score (BD 6, S1) et dans d'autres cas elle le dégrade (BD 1). De même pour le critère du nombre d'itérations.

Partant de cette constatation, on décide qu'il serait plus judicieux de compter le nombre de voisins générés durant chaque itération. Suite à cela, on affectera à la taille de la liste taboue, la racine carrée du nombre des voisins générés durant chaque itération de l'algorithme.

Grace à cette automatisation du choix de la taille de la liste taboue, l'algorithme sera capable de s'auto-fixer ce paramètre à chaque itération. Par ailleurs, dans le but d'ajouter de la

flexibilité à ce mécanisme, on propose de rajouter à la taille de la liste taboue, un nombre tiré aléatoirement et compris entre deux bornes qu'on fixera.

- **Sélection du paramètre P2**

Après un certain nombre de violations successives ou de respects successifs de la contrainte relative au coût, l'algorithme met à jour la valeur de β. Le choix de la nouvelle valeur qui sera affectée à β suite à des respects successifs de la contrainte modifie considérablement le score final. En effet, au cours de ce test, on a essayé de voir quelles conséquences auraient la mise à jour de β en la divisant par 2 versus en la divisant par un grand nombre 5000.

On découvre grâce au tableau 5.7, que la diminution de β lors de sa mise à jour dégrade complétement le score. Par conséquent, on décide de mettre à jour la valeur de β en la divisant par 2 à chaque mise à jour.

- **Sélection du paramètre P3**

Comme on ne peut pas laisser β croitre ou décroitre indéfiniment, on lui a fixé des bornes. Maintenant on cherche à savoir quelles seront les valeurs de ces bornes.

Au cours de ce test, on a augmenté la valeur de la borne supérieure de β. En analysant les résultats récupérés dans le tableau 5.7, on constate que pour certains exemplaires testés, il y a eu une amélioration du score. Cela dit, on note aussi qu'il peut aussi y avoir des dégradations du score dans d'autres exemplaires.

Comme solution à ce problème, on décide d'automatiser ce mécanisme et laisser l'algorithme auto-régler ce paramètre tout seul. Si au bout de 50 itérations successives, β est au seuil maximal, alors on double la valeur de la borne supérieure de β et vice-versa.

- **Sélection du paramètre P4**

Dans ce test, on essaie de trouver la bonne valeur du nombre d'itérations successives de respect de la contrainte ou de sa violation, avant de mettre à jour le paramètre β. En analysant les résultats du tableau 5.7, on voit qu'une augmentation de ce nombre n'est pas toujours une bonne action. Par contre, on arrive à bien voir que le nombre d'itérations pris par le programme pour arriver au même résultat que celui de P0 est plus grand. Ceci s'explique par le fait que le Tabou aura besoin de plus d'itérations pour mettre à jour β dans le cas où l'on fixe ce paramètre P4 à 500 au lieu de 10 dans la configuration de base.

On décide alors de changer la valeur de ce paramètre et de le forcer manuellement selon la progression de l'algorithme Tabou. Au début de la recherche, il est intéressant de constater que la valeur de P4 est assez grande le temps que tout le processus de réglage des autres mécanismes se stabilise et que le programme arrive à trouver des solutions d'une assez bonne qualité. Puis lorsque l'algorithme entre dans sa phase d'intensification, on réduit la valeur de P4 afin de permettre à β de se mettre à jour assez rapidement et ainsi de régler la direction de la recherche.

5.2 Résultats finaux du programme implémenté

5.2.1 Présentation des séries de tests

Dans cette section, on présente tous les résultats qui ont été obtenus par le module d'optimisation développé tout au long de cette maîtrise. Tous les résultats des tests présentés dans cette section ont été réalisés sur un ordinateur tournant sur un processeur Intel i5 cadencé à 1.70 GHz et 6 Go de RAM.

Dans un premier lieu, on présente dans des tableaux les résultats des tests effectués sur six différentes bases de données avec un modèle linéaire. Ensuite, dans une seconde étape on remplace notre fonction objectif linéaire par une fonction objectif quadratique puis on relance le module d'optimisation pour pouvoir récupérer les nouveaux résultats issus du modèle non linéaire.

Comme il est indiqué dans la section 4.3.1, la taille du voisinage est dix fois supérieure à *n*, ce qui ralentit considérablement l'évolution de la recherche surtout quand on dépasse les 1 000 mots clés. Afin d'accélérer l'évolution de la recherche, on implémente une méthode qui sélectionne aléatoirement, durant chaque itération, de 50 à 250 mots clés pour lesquels le voisinage sera généré. Avec cette manière de faire, le processus de recherche est beaucoup plus rapide que dans le cas où l'on considère la totalité des mots clés sans une grande dégradation de la qualité du score.

5.2.2 Résultats de la modélisation linéaire

Comme il est mentionné dans la section 4.1.2, on se propose de tester 3 différentes solutions initiales sur six différentes bases de données. Les résultats issus de chacune de ces trois solutions seront présentés dans un tableau. Pour le cas des deux premières solutions initiales, deux différentes approches ont été considérées. La première est appelée solution de type 1, elle

arrête la recherche dès qu'elle atteint 95% du score optimal donné par CPLEX, quant à la deuxième (solution type 2), elle n'arrête la recherche qu'après avoir atteint 200 000 itérations. Par contre, pour le cas de la troisième solution initiale on interrompt la recherche uniquement après 10 000 itérations successives sans amélioration du score de S* puisque la solution initiale utilisée dans ce cas dépasse déjà 99% le score optimal donné par CPLEX.

Par ailleurs, dans le but de vérifier la robustesse du module d'optimisation, on se propose d'exécuter pour chaque instance et pour chaque type de solution, 10 fois le programme développé avec une graine aléatoire différente. À la suite de chaque test, on récupère deux indicateurs de performances qui sont le score optimal atteint par le module d'optimisation ainsi que le temps d'exécution qu'il a fallu au module pour achever chaque test. Une fois que tous les tests ont été finalisés, on récupère toutes les valeurs des indicateurs pour en générer la valeur minimale, la valeur maximale, la moyenne et enfin l'écart type qu'on présente dans les tableaux 5.8, 5.9 et 5.10. Comme on peut le voir dans ces trois tableaux, on affiche aussi le taux de rapprochement du score de la solution issue du Tabou par rapport à score atteint par CPLEX. Étant donné que pour les solutions de type 1, l'objectif est d'atteindre 95% de la solution optimale de CPLEX, le taux n'est plus intéressant à présenter. Finalement, on rajoute dans le tableau 5.10 une ligne appelée score S_0 qui présente le score de la solution initiale S_0'''.

❖ Solution S_0'

Tableau 5.8: Résultats des tests du module d'optimisation issus de S_0'

BD 1 500 mots clés	Cplex		Score optimal	224,45			
			Temps d'exécution	2,33 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	213,23	213,34	213,25	0,04
			Temps d'exécution	8,33s	29,12s	15,87s	8,4s
		Solution type 2	Score optimal	217,29	224,07	221,10	2,52
			Taux	0,9681	0,9983	0,9851	
			Temps d'exécution	36m 51,7s	47m 49,34s	40m 56,76s	3m 42,27s
BD 2 500 mots clés	Cplex		Score optimal	3 940,48			
			Temps d'exécution	1,82 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	3 743,46	3 747,68	3 744,48	1,34
			Temps d'exécution	1m 1,58s	2m 17,73s	1m 34,76s	26,95s
		Solution type 2	Score optimal	3 832,39	3 878,13	3 855,30	15,67
			Taux	0,9726	0,9842	0,9784	
			Temps d'exécution	33m 50,34s	41m 33,25s	38m 20,67s	2m 36,9s
BD 3 1000 mots clés	Cplex		Score optimal	29 859,48			
			Temps d'exécution	2,36 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	28 366,55	28 384,02	28 370,68	5,86
			Temps d'exécution	36,1s	6m 27,36s	1m 24,95s	1m 47,98s
		Solution type 2	Score optimal	28 347,32	29 539,12	29 078,10	335,35
			Taux	0,9494	0,9893	0,9738	
			Temps d'exécution	7m 31,43s	59m 28,34s	33m 2,62s	22m 4,07s
BD 4 1000 mots clés	Cplex		Score optimal	19 731,42			
			Temps d'exécution	3,27 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	18 664,26	18 760,33	18 740,47	27,41
			Temps d'exécution	21,45s	8m 45,81s	2m 41,01s	2m 36,08s
		Solution type 2	Score optimal	18 698,18	19 278,38	19 021,16	196,74
			Taux	0,9476	0,9770	0,9640	
			Temps d'exécution	46m 23,75s	1h 0m 26,79s	54m 40,27s	5m 3,07s
BD 5 5000 mots clés	Cplex		Score optimal	27 221,60			
			Temps d'exécution	9,18 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	25 860,65	25 871,14	25 864,47	3,48
			Temps d'exécution	12m 51,19s	32m 57,16s	19m 57,99s	6m 45,55s
		Solution type 2	Score optimal	25 544,90	27 192,20	26 369,67	551,64
			Taux	0,9384	0,9989	0,9687	
			Temps d'exécution	1h 11m 21,99s	3h 20m 48,18s	2h 14m 8,71s	41m 14,12s
BD 6 5000 mots clés	Cplex		Score optimal	2 392,84			
			Temps d'exécution	9,81 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	2 273,19	2 273,73	2 273,30	0,19
			Temps d'exécution	4m 17,72s	16m 35,96s	8m 40s	4m 51,07s
		Solution type 2	Score optimal	2 280,39	2 387,82	2 334,82	32,29
			Taux	0,9530	0,9979	0,9758	
			Temps d'exécution	45m 59,88s	3h 7m 56,49s	2h 12m 39,62s	43m 49,35s

Comme on peut le constater à partir du tableau 5.8, le module d'optimisation ne parvient pas à surpasser les performances de CPLEX. En effet, d'un côté les temps d'exécution de CPLEX sont plus petits que ceux du Tabou. Et d'un autre côté, les scores optimaux atteints par CPLEX dépassent ceux du Tabou. Bien que les performances de CPLEX soient meilleures que celles du Tabou, les résultats donnés par le module d'optimisation restent raisonnables. En effet, dans le cas des solutions de type 2, les scores optimaux atteints par notre module frôlent le score optimal de CPLEX. Cependant, il est vrai que leur temps d'exécution est parfois inacceptable.

Comme le montre la figure 4.8, l'évolution du score de S* est plus rapide au début de l'exécution qu'à sa fin. Cette constatation est aussi révélée à partir du tableau 5.8. En effet, on remarque que le Tabou prend beaucoup moins de temps pour atteindre 95% du score optimal que le temps qu'il lui faut pour améliorer le score de S* au-dessus de ce seuil. On donne comme exemple la sixième base de données. Il a fallu en moyenne 8 minutes et 40 secondes pour que le Tabou atteigne 95% du score de CPLEX. Par contre, il lui a fallu en moyenne plus que 2 heures pour atteindre 97.58% du score optimal.

Étant donné que le fait d'atteindre le seuil de 95% du score optimal est jugé comme étant un résultat d'une bonne qualité et que le temps de traitement considéré comme un temps raisonnable pour atteindre ce seuil est de l'ordre d'une dizaine de minutes, on peut alors dire que les résultats atteints par la solution de type 1 avec comme solution initiale S_0' sont acceptables.

❖ Solution S_0''

Tableau 5.9 : Résultats des tests du module d'optimisation issus de S_0"

BD 1 500 mots clés	Cplex		Score optimal	224,45			
			temps d'exécution	2,33 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	213,23	213,43	213,28	0,07
			Temps d'exécution	6,51s	25,75s	15,08s	7,92s
		Solution type 2	Score optimal	219,19	223,64	222,21	1,23
			Taux	0,9766	0,9964	0,9900	
			Temps d'exécution	32m 25,15s	49m 51,78s	39m 23,89s	5m 29,23s
BD 2 500 mots clés	Cplex		Score optimal	3 940,48			
			temps d'exécution	1,82 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	3 743,47	3 765,38	3 748,08	6,84
			Temps d'exécution	27,83s	2m 33,94s	1m 25,97s	38,32s
		Solution type 2	Score optimal	3 812,84	3 888,65	3 844,33	25,98
			Taux	0,9676	0,9868	0,9756	
			Temps d'exécution	33m 16,13s	45m 1,74s	39m 27,45s	3m 39,53s
BD 3 1000 mots clés	Cplex		Score optimal	29 859,48			
			temps d'exécution	2,36 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	28 366,53	28 374,33	28 367,78	2,35
			Temps d'exécution	1m 25,47s	8m 40,81s	3m 58,18s	2m 43,03s
		Solution type 2	Score optimal	28 540,84	29 348,43	28 840,82	308,56
			Taux	0,9558	0,9829	0,9659	
			Temps d'exécution	7m 37,01s	52m 44,72s	20m 36,02s	16m 30,24s
BD 4 1000 mots clés	Cplex		Score optimal	19 731,42			
			temps d'exécution	3,27 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	18 744,95	18 752,21	18 747,45	2,47
			Temps d'exécution	36,33s	9m 10,71s	2m 10,34s	2m 44,76s
		Solution type 2	Score optimal	18 866,09	19 380,06	19 132,59	208,17
			Taux	0,9561	0,9822	0,9697	
			Temps d'exécution	49m 51,6s	1h 10m 35,75s	1h 0m 26,41s	7m 8,12s
BD 5 5000 mots clés	Cplex		Score optimal	27 221,60			
			temps d'exécution	9,18 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	25 860,54	25 872,93	25 863,25	3,87
			Temps d'exécution	9m 9,8s	27m 1,77s	14m 10,72s	5m 3,91s
		Solution type 2	Score optimal	25 933,85	27 147,01	26 640,87	447,32
			Taux	0,9527	0,9973	0,9787	
			Temps d'exécution	1h 0m 7,09s	3h 11m 39,23s	1h 44m 17,69s	46m 31,36s
BD 6 5000 mots clés	Cplex		Score optimal	2 392,84			
			temps d'exécution	9,81 s			
				Min	Max	Moyenne	Ecart type
	Tabou	Solution type 1	Score optimal	2 273,19	2 274,62	2 273,40	0,45
			Temps d'exécution	4m 45,86s	26m 33,27s	11m 1,09s	6m 24,4s
		Solution type 2	Score optimal	2 295,22	2 357,71	2 337,80	23,38
			Taux	0,9592	0,9853	0,9770	
			Temps d'exécution	1h 0m 55,84s	2h 25m 28,07s	1h 16m 37,71s	32m 11,45s

Les résultats présentés dans le tableau 5.9 sont généralement de meilleure qualité que ceux présentés dans le tableau 5.8. Par exemple, dans le cas de la cinquième base de données, la moyenne des scores optimaux qu'affiche S_0" est plus grande que la moyenne des score de S* atteinte à partir de S_0'. De même pour les temps d'exécution. Bien que la différence entre les résultats obtenus à partir de la solution initiale S_0" et ceux obtenus de S_0' n'est pas si grande, on peut affirmer que la qualité de la solution initiale influe sur les performances du module d'optimisation.

Même si les résultats obtenus de S_0" sont généralement meilleurs que ceux issus de la solution S_0', les résultats donnés par CPLEX restent toujours les meilleurs.

❖ Solution S_0’’’

Tableau 5.10 : Résultats des tests du module d'optimisation issus de S_0'''

BD 1 500 mots clés	Cplex	Score optimal	224,45			
		Temps d'exécution	2,33 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	223,97			
		Score optimal	224,37	224,41	224,39	0,01
		Taux	0,9996	0,9998	0,9997	
		Temps d'exécution	23,78s	46,02s	32,8s	7,59s
BD 2 500 mots clés	Cplex	Score optimal	3 940,49			
		temps d'exécution	1,82 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	3 936,69			
		Score optimal	3 940,27	3 940,49	3 940,42	0,07
		Taux	0,9999	1,0000	0,9999	
		Temps d'exécution	19,31s	43,97s	27,72s	7,07s
BD 3 1000 mots clés	Cplex	Score optimal	29 859,49			
		Temps d'exécution	2,36 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	29 857,70			
		Score optimal	29 859,23	29 859,49	29 859,43	0,10
		Taux	0,9999	1,0000	0,9999	
		Temps d'exécution	24,23s	52,84s	34,42s	8,52s
BD 4 1000 mots clés	Cplex	Score optimal	19 731,42			
		Temps d'exécution	3,27 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	19 675,30			
		Score optimal	19 720,80	19 729,13	19 726,10	2,65
		Taux	0,9995	0,9999	0,9997	
		Temps d'exécution	29,97s	1m 29,79s	1m 7,87s	17,86s
BD 5 5000 mots clés	Cplex	Score optimal	27 221,60			
		Temps d'exécution	9,18 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	27 211,80			
		Score optimal	27 221,56	27 221,60	27 221,59	0,02
		Taux	0,9999	1,0000	0,9999	
		Temps d'exécution	6m 30,74s	15m 35,05s	10m 11,49s	2m 52,92s
BD 6 5000 mots clés	Cplex	Score optimal	2 392,84			
		Temps d'exécution	9,81 s			
	Tabou		Min	Max	Moyenne	Ecart type
		Score initial	2 392,63			
		Score optimal	2 392,81	2 392,83	2 392,82	0,01
		Taux	0,9999	0,9999	0,9999	
		Temps d'exécution	5m 27,06s	16m 35,94s	9m 37,5s	3m 10,88s

Dans le tableau 5.10, on remarque à partir des lignes présentant le score initial issu de la relaxation linéaire, qu'il est déjà très proche du score optimal de CPLEX. Malgré ce rapprochement, notre module d'optimisation parvient à améliorer encore le score comme le montre par exemple les tests de la quatrième base de données. En effet, le score initiale $S_0^{'''}$ de la base de données numéro 4 était 19.675,3 et il croît à une moyenne de 19.726,1 après l'exécution du module d'optimisation.

Par ailleurs, comme on peut le constater à partir des tableaux 5.8, 5.9 et 5.10, on note généralement l'existence d'un petit écart type ce qui nous permet de dire que notre module d'optimisation est assez robuste.

En comparant les résultats présentés dans le tableau 5.10 avec les deux autres tableaux 5.8 ou le tableau 5.9, on constate que notre module d'optimisation présente de meilleurs résultats lors de son départ de $S_0^{'''}$. En effet, ce résultat est attendu vu que $S_0^{'''}$ fournit déjà un score à 99% du score optimal.

5.2.3 Résultats de la modélisation non linéaire

Dans cette section, on teste notre module d'optimisation dans le cas où la fonction objectif est quadratique. Dans les résultats (sous forme de tableaux) qui suivent, on compare toujours les résultats issus du programme développé avec les résultats optimaux donnés par CPLEX. Mais lors de la présence de la forme quadratique, CPLEX peine à trouver la solution optimale. En effet, nous avons constaté que juste avec une vingtaine de mots clés il a besoin de plus d'une heure pour trouver la solution optimale. L'étude de la convexité de notre fonction quadratique confirme qu'elle est convexe puisque son Hessien est semi-défini positif. Donc CPLEX devrait être capable de résoudre ce genre de problème. Mais étant donné que CPLEX prend énormément de temps pour trouver la solution optimale, on comprend alors que malgré le fait que le problème soit artificiel, il reste tout de même difficile à résoudre.

Ainsi afin de pouvoir comparer les résultats du Tabou à ceux de CPLEX, les résultats des tests initiaux qui seront présentés dans le tableau 5.11 se limiteront à une quinzaine de mots clés, vu les limitations que CPLEX impose.

Tableau 5.11 : Résultats des tests du module d'optimisation avec une fonction objectif quadratique

BD	Num	1	2	3
	Nb mots clés	15	10	12
CPLEX	Score optimal	544 357,142	27 418 485,347	7 949 785 505,890
	Temps d'exécution	12 m 50,29 s	6 m 11,16 s	14 m 02,1 s
Tabou	Score optimal	543 576,290	27 219 731,620	7 894 006 175,000
	Taux	0,999	0,993	0,993
	Nb itérations	13 638	16 944	13 472
	Temps d'exécution	37,791 s	44,631 s	37,912 s

Comme nous pouvons le remarquer d'après le tableau 5.11, dès l'apparition de la forme quadratique dans la fonction objectif, CPLEX peine pour trouver la solution optimale. Avec moins de 15 mots clés, CPLEX prend énormément de temps de traitement pour fournir sa meilleure solution. Par contre, dans la même situation, notre module d'optimisation parvient en quelques secondes à atteindre 99 % du score obtenu par CPLEX.

Dans l'optique de comparer les résultats du Tabou avec des bases de données plus grandes, on développe un programme à base de l'algorithme Glouton. Cet algorithme choisit au hasard un ordre de mots clés durant chaque itération. Ensuite il essaie d'affecter la meilleure position à l'ensemble des *n* mots clés, tant que les contraintes du problème sont satisfaites.

Afin de comparer les résultats du Tabou avec le Glouton, on laisse le programme du Glouton s'exécuter durant le même temps qui est pris en moyenne par l'algorithme Tabou. Comme critère d'arrêt pour le Tabou, on a choisi de le stopper après 10.000 itérations sans amélioration du score de S*.

Comme pour les tests réalisés à la section 5.2.2, les tests finaux de cette section sont effectués sur les mêmes bases de données et chaque test est répété 10 fois avec des graines de nombres aléatoires différentes. On présente aussi le taux de rapprochement mais au lieu qu'il soit calculé par rapport au score optimal de CPLEX, il est comparé au score obtenu par l'algorithme Glouton.

Tableau 5.12 : Résultats finaux des tests du module d'optimisation dans le cas quadratique

			Min	Max	Moyenne	Ecart type
BD 1 500 mots clés	Glouton	Score optimal	635 217,96	688 605,75	662 131,53	19 767,52
		Temps d'exécution	5m 23,14s			0
	Tabou	Score optimal	1 572 940,15	1 703 836,43	1 627 272,40	57 376,61
		Taux	2,48	2,47	2,46	
		Temps d'exécution	5m 15,46s	5m 35,29s	5m 23,14s	6,95s
BD 2 500 mots clés	Glouton	Score optimal	1 043 862 387	1 215 142 259	1 115 230 900,3	49 129 181,87
		Temps d'exécution	5m 28,27s			0
	Tabou	Score optimal	2 837 493 940	2 973 167 679	2 911 551 370	44 477 668,44
		Taux	2,72	2,45	2,61	
		Temps d'exécution	5m 24,94s	5m 33,37s	5m 28,27s	2,58s
BD 3 1000 mots clés	Glouton	Score optimal	3 269 062 273	3 540 629 789	3 427 585 661,5	97 884 695,80
		Temps d'exécution	8m 29,82s			0
	Tabou	Score optimal	23 064 641 460	23 897 998 020	23 524 254 008	283 159 089,49
		Taux	7,06	6,75	6,86	
		Temps d'exécution	8m 17,12s	8m 45,58s	8m 29,82s	11,52s
BD 4 1000 mots clés	Glouton	Score optimal	20 146 794 280	26 659 227 190	22 431 961 715	1 976 294 948,37
		Temps d'exécution	8m 43,08s			0
	Tabou	Score optimal	76 050 008 020	78 414 615 550	77 542 788 934	855 084 022,53
		Taux	3,77	2,94	3,46	
		Temps d'exécution	8m 23,78s	9m 6,38s	8m 43,08s	16,83s
BD 5 5000 mots clés	Glouton	Score optimal	10 136 248 200	14 029 826 320	13 349 973 695	1 156 766 412,57
		Temps d'exécution	59m 24,23s			0
	Tabou	Score optimal	92 920 251 580	102 624 558 700	96 991 355 291	2 644 252 489,45
		Taux	9,17	7,31	7,27	
		Temps d'exécution	57m 57,77s	1h 2m 51,7s	59m 24,23s	1m 34,41s
BD 6 5000 mots clés	Glouton	Score optimal	57 182 933,22	79 256 724,99	63 136 552,89	6 804 034,59
		Temps d'exécution	57m 48,92s			0
	Tabou	Score optimal	621 499 120,40	681 040 564,70	641 792 927,36	20 528 800,81
		Taux	10,87	8,59	10,17	
		Temps d'exécution	57m 2,23s	58m 40,55s	57m 48,92s	35,08s

D’après les résultats présentés dans le tableau 5.12, on constate que notre module d’optimisation parvient à atteindre des résultats qui sont nettement meilleurs que ceux atteints par l’algorithme Glouton. En effet pour chacune des bases de données testées, l’algorithme Tabou présente des scores plus grands que ceux du Glouton. On remarque aussi, qu’au fur et à mesure que le nombre de mots clés des instances augmente, le taux augmente lui aussi. Prenant comme exemple les bases de données numéros 1, 3 et 6. Pour la première instance qui contient 500 mots clés, le taux indique que la moyenne des scores optimaux du Tabou est 2,46 fois plus grande que la moyenne des scores du Glouton. Quant à la troisième base de données formée de 1 000 mots clés, elle affiche un taux de 6,86. Finalement, avec 5 000 mots clés, la base de données numéro 6 présente un taux de 10,17. Donc plus les instances sont grandes, plus les performances de notre module d’optimisation sont meilleures comparées à celles de l’algorithme Glouton.

D’après les différents résultats issus du module d’optimisation développé au cours de cette maîtrise, on constate que l’algorithme Tabou est très performant du point de vue de qualité de la solution surtout dans le cas de la résolution d’un problème non linéaire. Même si CPLEX présente de meilleurs résultats que le Tabou lors de la résolution des problèmes linéaires, les résultats fournis par le Tabou restent acceptables.

Dans ce chapitre, on a commencé par décrire tout le processus de paramétrage de l’algorithme Tabou. Une fois que la version finale du module d’optimisation est mise au point, on a présenté et analysé les résultats finaux de notre module. Grâce à ces analyses, on a pu démontrer l’efficacité et la flexibilité de l’algorithme Tabou face aux outils d’optimisation commerciaux. Hormis le fait que notre module atteigne aisément 95% de la solution optimale dans de cas des problèmes linéaires, ce module permet aussi d’atteindre 99% de la solution optimale dans le cas du modèle non linaire en un temps au moins huit fois plus court que celui de CPLEX.

CONCLUSION

Ce projet de maîtrise a proposé une nouvelle approche pour l'optimisation de la gestion des campagnes publicitaires dans les moteurs de recherches d'Internet. Nous avons développé et implanté un module d'optimisation à base de l'algorithme Tabou permettant de maximiser le rendement de toute une campagne publicitaire sur Internet.

Nous avons tout d'abord fait une revue bibliographique qui nous a permis de déceler que l'optimisation des compagnes publicitaires sur Internet est un axe de recherche récent. Ainsi nous avons pensé à adopter la recherche Tabou puisqu'elle s'est révélée d'une grande efficacité lors de son utilisation dans plusieurs autres domaines. L'implémentation de cet algorithme nous a permis de nous affranchir des limitations des méthodes d'optimisation déterministes. Ces méthodes généralement implantées dans les outils commerciaux présentent des limitations lors de la présence de la forme quadratique. En réalisant des tests sur six différents échantillons de la base de données de notre partenaire industriel, nous avons constaté que dans le cas linéaire, CPLEX atteint la solution optimale en un temps plus court que celui du Tabou. Mais tout de même, la solution finale générée par notre module atteint 95% de la solution optimale en un temps raisonnable, de l'ordre de quelques minutes. Ceci est vrai pour les trois solutions initiales testées. Cependant, les mêmes tests réalisés avec la forme quadratique, ont montré que CPLEX est incapable de rivaliser avec les performances de notre algorithme.

Ainsi, l'implémentation du Tabou dans notre module d'optimisation, nous a permis de résoudre le problème dépendamment de sa forme en un temps acceptable. De ce fait, notre partenaire industriel est satisfait de la qualité de la solution que nous lui proposons, d'autant plus qu'il ne privilégie pas l'utilisation des solveurs commerciaux.

Le but principal de ce travail était de montrer l'efficacité de la recherche Tabou pour l'optimisation des campagnes publicitaires sur Internet. On a aussi proposé des techniques d'intensifications et de diversification qui ont permis d'améliorer considérablement les performances de notre algorithme. Étant donné que le choix de la solution initiale avait une influence considérable sur la qualité des résultats, il serait intéressant d'investiguer ce volet. Dans une perspective plus large, on propose d'étendre le module actuel afin de couvrir d'autres canaux de publicités sur Internet tels que les réseaux sociaux et les bannières.

BIBLIOGRAPHIE

[1] "Internet", complément encyclopédique du Petit Larousse, 2014. [En ligne]. Disponible : http://www.larousse.fr/encyclopedie/divers/Internet/125060. [Consulté le 11 février 2014].

[2] Unité régionale de formation à l'information scientifique et technique de Bretagne et des Pays de la Loire, "Comment définir internet? L'impossible définition," Problèmes de définition d'internet. [En ligne]. Disponible: http://www.sites.univrennes2.fr/urfist/node/55#Commentdefinir. [Consulté le 11 février 2014].

[3] Autorité canadienne pour les enregistrements internet, "L'économie d'internet au Canada," 2012. [En ligne]. Disponible: http://www.cira.ca/factbook/internet_economy_fr.html. [Consulté le 11 février 2014].

[4] D. Silverman, "IAB internet advertising revenue report 2012 first six months' results October 2012," PwC PricewaterhouseCoopers et IAB Interactive Advertising Bureau, New York, États-Unis, Sondage industriel, 2012. [En ligne]. Disponible: http://www.iab.net/media/file/IAB_Internet_Advertising_Revenue_Report_HY_2012.pdf. [Consulté le 12 février 2014].

[5] G. Hervet, "Attention et évitement des bannières publicitaires sur internet : quelles conséquences?," Ph.D., Faculté des Sciences de L'administration Université Laval, QC, Canada, 2012. [En ligne]. Disponible: www.theses.ulaval.ca/2012/28529/28529.pdf. [Consulté le 12 février 2014].

[6] C.Y. Yoo, "Background and Web Ad Effects Model," in Preattentive Processing of Web Advertising, 2007, United States of America, Cambria Press, 2007. [En ligne]. Disponible: books.google.ca/books?isbn=1621968456. [Consulté le 12 février 2014].

[7] D. Silverman, "IAB internet advertising revenue report 2011 Full Year Results," PwC PricewaterhouseCoopers et IAB Interactive Advertising Bureau, New York, États-Unis, Sondage industriel, 2012. [En ligne]. Disponible: http://www.iab.net/media/file/IAB_Internet_Advertising_Revenue_Report_FY_2011.pdf. [Consulté le 12 février 2014].

[8] B. Bathelot, "Définition Moteur de recherche," Definition Web-Marketing, 2011. [En ligne]. Disponible: http://www.definitions-webmarketing.com/Definition-Moteur-de-recherche. [Consulté le 12 février 2014].

[9] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords," National Bureau of Economic Research 2005.

[10] L. Hou, L. Wang, and J. Yang, "Evolutionary prediction of online keywords bidding," in E-Commerce and Web Technologies, ed: Springer, 2008, pp. 124-133.

[11] A. Agarwal, K. Hosanagar et M. D. Smith, "Location, Location, Location: An Analysis of Profitability of Position in Online Advertising Markets," Heinz Research, paper 56, 2008, pp. 3. [En ligne]. Disponible: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1055&context=heinzworks&sei-redir=1%20-%20search=%22Location,+Location,+Location,+An+Analysis+of+Profitability+of+Position. [Consulté le 13 février 2014].

[12] Acquisio, "L'entreprise," 2003, [En ligne]. Disponible: http://fr.acquisio.com/?page_id=28. [Consulté le 13 février 2014].

[13] GNU, "GLPK - the GNU Linear Programming Kit," 2014, [En ligne]. Disponible: https://ftp.gnu.org/gnu/glpk. [Consulté le 3 septembre 2014].

[14] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press, 1975.

[15] C. Darwin, "On the origins of species by means of natural selection," *London: Murray*, 1859.

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, pp. 1087-1092, 1953.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simmulated annealing," *science*, vol. 220, pp. 671-680, 1983.

[18] M. M. Aldaihani and T. M. Aldeehani, "Portfolio optimization models and a tabu search algorithm for the Kuwait Stock Exchange," *Investment Management and Financial Innovations,* vol. 5, pp. 31-40, 2008.

[19] O. Marinoni, A. Higgins, and S. Hajkowicz, "Portfolio optimisation of water management investments," ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 423-437.

[20] F. Busetti, "Metaheuristic approaches to realistic portfolio optimization," arXiv preprint cond-mat/0501057, 2005.

[21] P. Quinn, "Modélisation et prédiction du comportement de mots clés dans des campagnes publicitaires sur les moteurs de recherche," M.Sc.A, École polytechnique de Montréal, QC, Canada, 2011. [En ligne]. Disponible: http://search.proquest.com/docview/1266232295 . [Consulté le 19 Avril 2014].

[22] B. J. Jansen and T. Mullen, "Sponsored search: an overview of the concept, history, and technology," *International Journal of Electronic Business,* vol. 6, pp. 114-131, 2008.

[23] Google Finance, Google Inc(NASDAQ:GOOG), Google Finance, 2014, [En ligne]. Disponible: https://www.google.com/finance?q=NASDAQ:GOOG&fstype=ii. [Consulté le 19 Avril 2014].

[24] Google Finance, Yahoo! Inc.(NASDAQ:YHOO), Google Finance, 2014, [En ligne]. Disponible: https://www.google.com/finance?q=NASDAQ:YHOO&fstype=ii . [Consulté le 19 Avril 2014].

[25] Google Finance, Baidu Inc (ADR)(NASDAQ:BIDU), Google Finance, 2014, [En ligne]. Disponible: https://www.google.com/finance?q=NASDAQ:BIDU&fstype=ii. [Consulté le 19 Avril 2014].

[26] F. GLOVER, "FUTURE PATHS FOR INTEGER PROGRAMMING AND LINKS TO ARTIFICIAL INTELLIGENCE," 1986.

[27] I. Charon, A. Germa et O. Hudry, "Méthodes approchées définies par un voisinage," in Méthodes d'optimisation combinatoire, Paris : Masson, 1996, pp. 165-185.

[28] F. Glover, "Tabu search-part I," *ORSA Journal on computing,* vol. 1, pp. 190-206, 1989.

[29] A. Hertz, "Métaheuristiques," Notes de cours, Théorie, Groupe d'études et de recherche en analyse des décisions GERAD. [En ligne]. Disponible: http://www.gerad.ca/~alainh/Metaheuristiques.pdf. [Consulté le 14 Mai 2014].

[30] F. Glover et M. Laguna, *Tabu search*: Springer, 1999.

[31] F. Glover, "Tabu search—part II," *ORSA Journal on computing,* vol. 2, pp. 4-32, 1990.

[32] Gendreau, M., Hertz, A., Laporte, G., 1994. «A Tabu Search Heuristic for the Vehicle Routing Problem». Management Science 40, pp. 1276-1290.

[33] C. Duhamel et A. Duhamel, "Introduction à CPLEX," 2009, [En ligne]. Disponible: http://www.isima.fr/~duhamel/ESDI/intro_CPLEX_v2.pdf. [Consulté le 3 septembre 2014].

[34] M. Bouchard, A. Hertz et G. Desaulniers, "Lower bounds and a tabu search algorithm for the minimum deficiency problem", Journal of Combinatorial Optimization, vol. 17, pp 168-191, 2009. [En ligne]. Disponible: http://link.springer.com/article/10.1007%2Fs10878-007-9106-0. [Consulté le 03 mars 2014].